I SEGRETI DEL 1541

commodore

EVM · COMPUTER

I SEGRETI DEL
1541

E.V.M. COMPUTERS

INTRODUZIONE

Questo manuale e' stato messo a punto per fornire all' utente che desideri programmare sulle unita' 1540/1541 tutte le informazioni possibili per ottenere il massimo dal proprio lavoro.
Molti utenti si limitano ad utilizzare l' unita' a dischi per immagazzinare semplicemente dei programmi o dei dati in forma elementare e quindi non hanno compreso che con il disco si possono realizzare una varieta' infinita di operazioni.
E' vero che il 1540/1541 e' probabilmente la piu' complessa delle unita' che l' utente acquista perche' comprende un coordinato HARDWARE, SOFTWARE e parti meccaniche che non sono, nei loro completi e complessi movimenti, di facilissima comprensione.
Ecco perche' non e' stato possibile raccogliere in un solo volume tutto quanto. Infatti, come ricorderemo piu' volte, ed in particolare per la gestione dei files di tutti i tipi e per la parte meccanica ed HARDWARE abbiamo pubblicato un altro volume : LE PERIFERICHE COMMODORE che si occupa anche delle stampanti e delle unita' a cassetta.
Il presente manuale vuole essere un complemento molto piu' specializzato del precedente citato.
Per questo motivo riportiamo invece che un' approfondita disamina dei comandi tutte le particolarita' connesse alla programmazione che abbiamo ritenuto importanti, il disassemblato del DOS, le prime pagine di memoria ed al termine una piccola serie di programmmi che riteniamo utilissimi.
Tutti i comandi inviati dall' unita' centrale al disco vengono esaminati e controllati dal DISK OPERATING SYSTEM appunto il DOS.

Il DOS e' contenuto nella memoria ROM del Floppy stesso. La versione attuale e' chiamata:

CBM DOS V2.6

Mentre ricordiamo che le unitaa' centrali COMMODORE 64 e VIC 20 sono equipaggiate con un sistema operativo comprendente anche il linguaggio Basic:

BASIC 2.0

L' accesso al DOS e' consentito ed esemplificato nel presente manuale con i comandi di accesso diretto.
Per ottenere pero' il meglio sarebbe opportuno utilizzare il linguaggio ASSEMBLER di cui fra non molto uscira' un corso:

CORSO DI ASSEMBLER II Ed. E.V.M.

Come sempre mettiamo a disposizione un dischetto, acquistabile separatamente o nell' apposita confezione che evita la noiosa operazione di digitazione dei programmi.
Questo nostro lavoro, come del resto quasi tutti i manuali del settore e per quanto attentamente corretto, non e' certamente privo di errori e refusi tipografici.
Per questo invitiamo i cortesi lettori a segnalarci quanto di inesatto troveranno.

CAPITOLO PRIMO

Con l' acquisto di un drive per floppy disk la potenzialita' operativa del vostro sistema e' fortemente aumentata.
L'uso piu' comune di un' unita' a dischi e' per l' immagazzinamento dei programmi o di dati. Immagazzinare programmi con un'unita' a dischi e' considerevolmente piu' facile che con un'unita' a cassette. Il piu' grande vantaggio di un'unita' a dischi e' la velocita' di trasferimento dati da e al computer. Vediamo un confronto:

Salvare un programma di 3 K BYTE richiede:

75 secondi con la cassetta
12 secondi con il disk drive.

Un altro vantaggio e' che un dischetto puo' immagazzinare molti piu' programmi che non una cassetta, anche se questo ragionamento, in particolare con l'unita' 1541, e' relativamente sfruttato.
Inoltre per caricare un programma si puo' consultare la Directory del disco per vedere quali programmi sono presenti sul dischetto stesso.
Infatti sebbene anche la cassetta consenta di immagazzinare piu' di un programma, la ricerca per ciascuno di essi e' molto piu' laboriosa e richiede una grande quantita' di tempo.
Molte di queste pagine provengono dal manuale LE PERIFERICHE COMMODORE Ed. EVM, al quale faremo spesso riferimento ed al quale rimandiamo per i seguenti argomenti:

-Trattamento dei files sequenziali e Random.

-Trattamento di TUTTI i tipi di periferiche, come stampanti,cassette, ecc.

-Notizie relative all' HARDWARE.

Molti concetti vengono anche ripetuti piu' volte, mentre altri sono dati per scontati, in particolare quelli relativi alle piu' comuni tecniche di programmazione che dovrebbero essere conosciute.
Infine i riferimenti, specie nella parte iniziale, alla cassettta sono fatti perche' la gran parte degli utenti prima di passare al disco inizia a lavorare con l' unita' a cassetta o DATASETTE.

FILES DISCO

I dischetti possono immagazzinare sia files programmi che files di dati. A differenza che sulle cassette, sui dischi si possono immagazzinare i files di dati in tre diversi modi, sempre sotto controllo di un programma:

FILES SEQUENZIALI

FILES RELATIVES

FILES RANDOM

I files relatives sono gestibili direttamente solo con le unita' che abbiano il DOS 4. Per gli utenti VIC e CBM64 abbiamo pero' messo a punto una routine che consente di utilizzarli comunque.

CONFRONTO FRA MANIPOLAZIONE DI FILES SU DISCO E SU CASSETTA.

La manipolazione di files su cassetta differisce in modo sostanziale da quella degli stessi files su disco per le seguenti ragioni:

1 - La velocita' di accesso ai dischetti e' molto piu' alta di quella delle cassette.

2 - Non esistono INIZI e FINE sulla superfice magnetica dei dischetti. Una unita' a dischi accede con facilita' a qualsiasi punto del disco cosa che ovviamente non avviene per la cassetta.

3 - La manipolazione di files di dati su cassetta o su disco differisce perche' la formattazione dei dati ed i metodi di accesso sono sostanzialmente diversi.

Non facciamoci ingannare dalla velocita' meccanica di rotazione del disco o della cassetta che se non e' uguale non determina comunque una grande differenza.
La cassetta registra i dati in maniera sequenziale durante lo scorrimento del nastro e nello stesso modo li rilegge.

Al contrario il disco immagazzina dati su un gran numero di tracce concentriche.
La testina di lettura della cassetta e' ferma ed aspetta che il nastro si posizioni per poter eseguire operazioni di lettura o scrittura, al contrario la testina dell' unita' a dischi si sposta avanti ed indietro mentre il dischetto gira per trovare il giusto punto in cui operare.

Per usare l' unita' a dischi non e' indispensabile sapere come le informazioni sono immagazzinate sulla superfice magnetica del supporto, tuttavia le conoscenze di questi argomenti renderanno piu' efficiente la programmazione.
Per questo inizieremo la nostra discussione sui files disco descrivendo il modo in cui i dati sono immagazzinati sulla superfice magnetica del floppy.

COME IL DISCO IMMAGAZZINA I DATI.

Un dischetto immagazzina i dati su un numero di tracce circolari concentriche.
Queste tracce sono a loro volta divise in settori.
Differenti unita' scrivono un diverso numero di tracce sul disco.
Alcuni drives scrivono su ambedue le facce del dischetto come gli 8250, gli altri come il 2040,3040,8050 e 1540/1541 su una sola.
I drives dell' unita' a dischi non scrivono dati lungo l' intera lunghezza della traccia, che e' utilizzata invece per la memorizzazione dei segnali di riferimento.
Per far questo sarebbe necessario un complesso sistema di indirizzamento.

Infatti se si utilizzasse l' intera superfice della traccia dovremo presumere che, dato che ci troviamo di fronte a tracce concentriche nessuna traccia avrebbe la lunghezza di un' altra e pertanto non potrebbe contenere la stessa quantita' di dati.
Per risolvere questo problema che inevitabilmente porterebbe ad un appesantimento eccessivo della gestione del Sistema Operativo su disco, le tracce sono state divise in settori.
Ogni settore contiene esattamente lo stesso ammontare di informazioni.
Nel caso dei dischi COMMODORE ogni settore contiene un blocco di dati pari a 256 Bytes.

DIRECTORY DEL DISCO E BAM

Due settori di ogni disco sono usate per l' indice del dischetto stesso.
Il primo.o DIRECTORY TRACK contiene il nome che e' stato assegnato al dischetto, seguito dalla BAM o BLOCK AVAILABILITY MAP che identifica i blocchi allocati cioe' gia' utilizzati per files di dati o di programmi oppure riservati con istruzioni di BLOCK-ALLOCATE.
Il secondo settore contiene i nomi di tutti i files presenti sul dischetto con l' indicazione del tipo di file ,la posizione del primo blocco utilizzato dal file, il nome del programma e la sua lunghezza.
Come abbiamo detto la BAM e' in pratica la rappresentazione della memoria disponibile su disco e della distribuzione degli spazi.
Quando il sistema deve immagazzinare dati su disco, la BAM viene automaticamente collegata con il DOS per determinare quale spazio e' disponibile e quindi quanti

blocchi possono essere salvati.
Se e' disponibile un spazio sufficiente per immagazzinare un dato file, allora l' operazione sara' coronata da successo e la BAM aggiornata per tener conto dello spazio utilizzato.

Se invece il DOS riterra' che lo spazio non e' sufficiente allora verra' riportato un errore e l' operazione stessa di salvataggio non avra' effetto e verra' solo registrato il nome del programma nella Directory con un asterisco.

Confronto con la cassetta

I files immagazzinati su cassetta non necessitano di una Directory o indice all' inizio del nastro.
Se dieci files sono immagazzinati su cassetta e il programma specifica un accesso particolare al sesto file, il fatto di avere una directory all' inizio del nastro non aiuta certo l' unita' a cassette a trovarlo meglio!
Poiche' un file su cassetta puo' avere una lunghezza qualsiasi, non esiste mezzo di associare il numero del file ad una determinata posizione del nastro.
Questo anche perche' non e' possibile avere dei comandi che facciano andare il nastro prima forte e poi piano senza usare i tasti dell' unita' a cassette.
Ne' per quanto utile, c' e' da fidarsi eccessivamente del contametri della DATASETTE tutto meno che preciso e che comunque non potrebbe tener conto dell' allungamento o dell' accorciamento del nastro dovuto alle variazioni termiche o al punto di scrittura.
Come abbiamo detto in precedenza l' unico sistema per

evitare di leggere i files che precedono quello oggetto della nostra ricerca e' di posizionarsi con il tasto FORWARD un po' prima dell' inizio (probabile) del file.
In caso contrario e' necessario far rileggere tutti i files precedenti.

Al contrario con una unita' a dischi si puo' andare direttamente all' inizio di un qualsiasi file sulla superfice del dischetto stesso, perche' ogni settore del disco e' egualmente accessibile.
Per rendere possibile questo e' quindi necessario che ogni dischetto abbia un indice che contenga il nome di tutti i files ivi registrati e l'indirizzo del settore di partenza.
Questo indice, simile quindi all' indice di un libro, e' appunto la DIRECTORY che fornisce anche il tipo di file che e' stato memorizzato e l' occupazione in blocchi di questo.

Quando un file di dati su disco e' aperto, l' unita' prima di tutto leggera' la Directory dalla quale ottiene l' indirizzo del settore in cui ha inizio il file. Poi la testina di lettura/scrittura potra' posizionarsi direttamente all' inizio del file aperto.

La Directory contiene le seguenti informazioni:

-Nome del disco

-Identificatore (ID) del disco

-Numero di versione del DOS

-Nome dei Files

-Tipo dei Files

-Numero dei blocchi usati

-Puntatore al primo blocco dei Files

-Numero dei blocchi disponibili

Spiegheremo approfonditamente nel resto del manuale.
Vediamo ora come vengono trattati i records di un file su disco.

FILES RELATIVES

Tutti i records presenti in un file relative hanno la stessa lunghezza.
Per questo e' facile calcolare l' indirizzo di settore per un singolo record di un file relative.
Supponiamo di avere un file relative in cui i singoli records occupino mezzo settore. Cioe' che ne entri due per settore.
Allora il decimo record di questo file relative sara' semplicemente rintracciabile sul quinto settore dall' inizio del file.

!!!ATTENZIONE!!!

I files relatives sono disponibili direttamente solo con le CBM BASIC VERSION 4.0 e oltre, usando il DOS 2.0 e

oltre.
Come abbiamo detto in questo manuale e' riportata una subrotine che consentira' anche agli utenti del VIC-20 e CBM64 di utilizzare i Files Relatives oltre ad altre particolarita' su questi files.

FILES SEQUENZIALI

I records di un file sequenziale possono avere differenti lunghezze.
Per questo non si puo' calcolare il settore sul quale deve essere rintracciato un particolare record di un file sequenziale, appunto perche' la lunghezza del singolo record e' sconosciuta.
La testina del dischetto puo' andare direttamente all' inizio di un file sequenziale, poiche' l' indirizzo del settore e' dato dalla Directory, ma una volta trovato questo inizio il file deve essere letto fino a quando non si trova il record desiderato.
La ricerca e' quindi sequenziale e quindi simile a quella su nastro.
Per trovare il decimo record di un file e' quindi necessario leggere i precedenti 9 records.

NOTA

Tutte le versioni dei dischi della Commodore sono abilitate alla gestione dei files sequenziali.

FILES SEQUENZIALI E RELATIVES

Se i records di un file dati sequenziale devono essere letti sequenzialmente cioe' uno dopo l' altro, gran parte dei vantaggi dell' accesso casuale tipico dei dischi viene perso. Ed allora perche' usarli?.
Prima di tutto perche' non su tutte le unita' a dischi della Commodore e' possibile la gestione dei FILES Relatives.
Secondo, e molto piu' importante e' che con i files sequenziali si riesce ad immagazzinare molte piu' informazioni che con i relatives. Si immagazzinano in forma piu' densa.
Si sfrutta cioe' meglio la capacita' di memorizzazione del disco.
Consideriamo il seguente esempio.
Si abbiano due nomi ed indirizzi come segue:

ALESSANDRO MARCELLOZZI
VIA MARTIRI DELLA LIBERAZIONE 7
20036 MILANO

e

MARIO ROSSI
VIA ROMA 1
31073 ROMA

Supponiamo che questi due nomi ed indirizzi facciano parte di un file MAILING LIS termine che troverete spesso per indicare programmi che gestiscono insiemi piu' o meno dettagliati di indirizzi con tecniche variamente sofisticate.
Ogni nome ed indirizzo diverra' quindi un record entro

il file dati.
Gestendo il file con il metodo relative si dovra' assegnare lo stesso spazio per ogni nome ed indirizzo.
Per questo si dovra' tenere conto non della media di occupazione ma dell' indirizzo piu' lungo. Di conseguenza i nomi e gli indirizzi piu' corti lasceranno parte del disco inutilizzato.
Nell' esempio precedente, per immagazzinare il primo nome ed indirizzo avremo bisogno di 69 bytes (considerando gli spazi di separazione) mentre per il secondo solo di 34, e quindi circa la meta'.
Ma usando un file relative dovremo dimensionarlo a records di 69 almeno per cui il secondo nominativo lascera' meta' del suo spazio inutilizzato.
Al contrario un file sequenziale assegna ad ogni record solo lo spazio che effettivamente gli necessita.
Per cui lo sfruttamento e' massimo e su grandi quantita' di dati si fa indubbiamente sentire.

INDIRIZZAMENTO DEL DISCO

I settori assegnati su disco ad un file di dati non sono FISICAMENTE sequenziali sulla superfice del dischetto anche quando si utilizza un file di tipo SEQUENZIALE.
Per esempio, quando si aggiungono records ad un file esistente, questi devono essere registrati senza andare a cadere sul file successivo.
Per questo il file dovra' essere proseguito, in casi di aggiunte, dovunque esistano settori liberi sulla superfice del dischetto.
Il file si contrae quando si cancellano records per cui, con questa operazione si rendono disponibili nuovamente dei settori precedentemente allocati.

Alla funzione di distribuzione del file sulla superfice del disco e' preposto il DOS cioe' Disk Operating System per cui la distribuzione su tutta la superfice del disco non presenta nessun problema quando si lavora con i files sequenziali.
Come vedremo in maniera in modo piu' approfondito nel seguito,e' presente un puntatore in ogni settore che dice in pratica dove indirizzarsi per la successiva lettura o scrittura.

APERTURA DI UN FILE SU DISCO

Solo 5 buffers di memoria sono disponibili su ogni unita' a disco 1541 per la manipolazione dei files.
Non appena si accede ad ogni file disco due di questi buffers sono usati per operazioni di controllo.
Cio' lascia 3 buffers in ogni unita' attraverso cui si puo' accedere ai files data stessi.
Vedremo poi le necessita' di buffers per ogni tipo di files.

INDIRIZZI SECONDARI

Il Basic usa 16 indirizzi secondari: da 0 a 15. Ogni comando di OPEN nel Basic deve specificare un indirizzo secondario.Gli indirizzi secondari sono usati nella seguente maniera:

1-L' indirizzo secondario 0 e' usato per caricare i

programmi dal disco alla memoria centrale del computer.

2-L' indirizzo 1 e' usato per salvare i programmi dalla memoria centrale del computer all' unita' a disco.

3-Gli indirizzi secondari da 2 a 14 sono usati per accedere ai files di dati (sono appunto 13 come ricordavamo prima).Si puo' selezionare uno qualunque di questi indirizzi secondari, ricordando che pero' poi non possono essere usati per un' altra operazione di OPEN su altro file di dati.

4-L' indirizzo secondario numero 15 apre uno speciale " CANALE DI COMANDO" che e' usato per accedere allo STATUS del dischetto e per consentire una delle speciali operazioni che vedremo successivamente.

IL CANALE DI COMANDO (15)

Il canale di comando necessita di una particolare attenzione perche' e' veramente molto importante.
Usando il disco del VIC, come del resto quelli della serie 3000, si dovrebbe sempre aprire il canale di comando per effettuare una qualsiasi operazione su disco.
Si dovrebbe inoltre lasciare questo canale aperto fino a quando si operi comunque ed in qualsiasi modo su disco.
Inoltre come abbiamo detto si usa il canale di comando per le operazioni speciali su disco e per interrogarlo sullo STATUS.

FORMATTAZIONE DI DISCHETTI NUOVI

Prima di adoperare un dischetto nuovo questi deve essere preparato. Questo sistema si chiama FORMATTAZIONE.
Che cosa s'intende per formattazione?
Ogni unita' a dischi' ha una sua speciale caratteristica. Un dischetto e' diviso in tracce e la informazioni vi vengono scritte su ogni traccia in maniera simile a come la musica viene registrata sui dischi per i fonografi. Il numero di tracce per dischetto varia da un fabbricante all'altro.
Ogni traccia e' divisa in settori il cui numero puo' anche esso variare. Durante il processo di formattazione vengono scritti sul dischetto dei settori vuoti. Un settore scritto in ogni traccia e indirizzi, e questi settori stessi sono riconosciuti tramite indirizzi univoci. Cio' consente al sistema operativo su disco (DOS) di identificare le singole posizioni su dischetto. Ad ogni settore viene dato un codice cosi' che il sistema operativo su disco possa riconoscere se questo dischetto e' stato formattato proprio con quel tipo di unita' a dischi. Il codice di formattazione per il 1541e' 2A.

NOTA

Per un approfondimento su questa operazione vedere il volume LE PERIFERICHE COMMODORE.

Ogni settore e' suddiviso in blocchi ognuno dei quali puo' contenere 256 caratteri.
Il compito finale della formattazione e' di costruire la directory del dischetto. La directory e' un indice di tutti i files contenuti nel dischetto. E' presente inoltre uno speciale insieme o blocco di dati chiamato BAM cioe' Block Availability Map che indica se un dato

blocco, che ricordiamo di 256 caratteri, su un dischetto e' gia' stato utilizzato oppure se ancora disponibile. La Directory e la Bam si possono ritrovare sulla traccia 18 del dischetto.

CAPITOLO SECONDO

OPERAZIONI SU DISCO

In aggiunta alle operazioni di scrittura e lettura files su dischetto che vedremo separatamente e dettagliatamente per ogni tipo di accesso , il Basic della Commodore relativo a questo tipo di unita' consente le seguenti operazioni:

1-Preparazione di un nuovo dischetto.

2-Cancellazione di un disco vecchio e preparazione per un nuovo uso.

3-Visualizzazione della Directory del disco per vedere quali file sono immagazzinati, quanto spazio questi hanno occupato e quindi quanto ne resta utilizzabile.

4-Copia di un file

5-Copia di un intero dischetto

6-Cancellazione di un file o rimpiazzo dei files

PREPARAZIONE DI UN DISCO E INIZIALIZZAZIONE

A differenza di quanto avviene per la cassetta non si puo' prendere un dischetto vergine, inserirlo nel drive ed incominciare a scrivere i dati.
Per prima cosa infatti la superfice magnetica deve essere preparata ad accogliere i dati, i settori devono essere fissati e poi devono essere scritte la Directory e la BAM.
Inoltre al dischetto deve essere assegnato un nome.

Si possono usare i dischi di una marca che si desidera, ma la casa che li costruisce non sa a priori se saranno adoperati su una marca di computer invece che su un' altra e all' interno della stessa marca essi possono venire utilizzati per un numero diverso di traccce e settori, ecco il motivo della preparazione o meglio della loro IDENTIFICAZIONE.

Inoltre si puo' ripreparare per un nuovo uso un vecchio dischetto, naturalmente purche' sia in condizioni fisiche integre e sopratutto non sia rigato.
Questa operazione cancella naturalmente tutti i dati vecchi, compresa la BAM e la Directory.

Di norma la preparazione di un dischetto per il suo uso viene fatta in modo diretto, anche se questa routine puo' essere inserita in un menu' di programma.

PREPARAZIONE

Per preparare un dischetto si deve per prima cosa eseguire un OPEN sul canale di comando.
Poi si eseguira' un comando PRINT# usando il file logico specificato nella lista dei parametri del comando OPEN.
Il comando PRINT# deve avere la seguente lista di caratteri, o parametri, racchiusa fra virgolette:

NEW o N

che identifica appunto l' operazione da eseguire.

: (due punti)

di separazione

NOME DEL DISCO

un nome qualsiasi che vogliamo dare

, (virgola)

anche questa di separazione

XX

identificatore del disco

E quindi il formato generale del comando che segue il PRINT# sara':

"NEW:NOME DEL DISCO,XX"

NOTA

NEW puo' essere rimpiazzato o abbreviato con la sola lettera N.
Il nome del disco deve essere una stringa di lunghezza non superiore ai 16 caratteri.
XX deve essere un coppia di caratteri alfanumerici.

Nel comando OPEN con il quale si accede al canale di comando si puo' mettere un qualsiasi numero di file logico, ma si deve specificare che l' unita' fisica e' la numero 8 e l' indirizzo secondario che deve essere il numero 15.
Il comando NEW viene usato su un dischetto non FORMATTATO oppure su un dischetto che l' utente vuole riformattare e del quale quindi non interessano piu' i dati.
Quando si usa il modo RIFORMATTAZIONE di un disco vecchio sara' cancellata la Directory preesistente e reinizializzata la BAM rendendo quindi nuovamente disponibili tutti i blocchi del dischetto.
In questo caso non dovremo specificare XX cioe' l' identificatore.
Vediamo qualche esempio.

```
OPEN 1,8,15
```

```
PRINT#1,"NEW:ESEMPIO,10"
```

RISULTATO: Viene aperto il canale di comando, formattato un disco che avra' per nome ESEMPIO e per identificatore 01.

Esempio:

```
OPEN3,8,15

PRINT#3,"N:TEST,11"
```

RISULTATO: Come il precedente ma con un disco di nome TEST e identificatore 11.

Riportiamo inoltre un esempio di riformattazione di un dischetto usato.

```
OPEN15,8,15

PRINT#15,"N:NUOVO"
```

RISULTATO: Con questo comando al disco usato viene assegnato un nome NUOVO, la Directory e la BAM vengono riformattate e resta solo l' identificatore preesistente.

NOTA

E' bene ricordare che questa ultima procedura potra' funzionare SOLO se il dischetto e' gia' stato formattato in precedenza e su quello stesso tipo di unita'.
Il tempo necessario per formattare un disco e' di circa 2 minuti.
Molto inferiore e' invece l' operazione di riformattazione.

NOTA

Il comando NEW al disco, che infatti viene dato fra virgolette non deve essere confuso con il NEW del BASIC che cancella il programma in memoria ed azzera le variabili.

ERRORI

Se per una qualsiasi ragione il disco non puo' essere formattato allora il LED rosso presente sulla parte frontale del drive, iniziera' a lampeggiare (nelle unita' a doppio floppy si accende solo la luce rossa).

La ragione puo' essere una delle seguenti:

1 - Si e' dimenticato di inserire il disco nell' unita' oppure si e' dimenticato di chiudere le alette o si e' inserito il disco al rovescio.
A questo proposito e' bene ricordare che il metodo di aprire un' altra finestrella per poter utilizzare il

dischetto da tutte e due le parti e' pratica assolutamente da evitare.

2-I comandi non sono stati dati correttamente.
E' bene porre particolare attenzione alla punteggiatura che e' l' errore in cui si cade piu' di frequente. Essa infatti deve essere SCRUPOLOSAMENTE osservata nelle sue regole.

3-Il disco ha la fascetta di protezione.

4-Il dischetto e' realmente difettoso.

5-L' unita'a a dischi e' difettosa.

NOTA

Sara' bene vedere in fondo al manuale le tavole di errore per un maggior approfondimento.

INIZIALIZZAZIONE

Benche' non sia indispensabile questa funzione sul drive 1540 o 1541 puo' accadere talvolta di doverla usare.
Per inizializzare il dischetto e' necessario eseguire un OPEN sul canale di comando e successivamente un comando di PRINT# seguito dalle parole fra virgolette "INITIALIZE" o dalla letterea "I".
Il comando "I" allinea la testina di lettura/scrittura con la traccia 1 del dischetto.
Successivamente la testina si posiziona sulla traccia 18, legge il nome del disco ed il suo identificatore e

carica queste informazioni nella memoria del Disk Operating System.
I dischi sono normalmente inizializzati in modo programma e nessun dato sulla superfice del disco e' variata durante questa operazione che quindi puo' anche avvenire con la finestrella coperta.

Esempio

```
10 OPEN1,8,15
20 PRINT#1,"INITIALIZE"
```

oppure

```
10 OPEN1,8,15
20 PRINT#1,"I"
```

Si puo' anche usare la forma abbreviata:

```
OPEN1,8,15,"I"
```

che si adopera in forma diretta quando si inizia ad operare su disco.

VALIDATE

Dopo che un dischetto e' stato usato per molto tempo, puo' succedere che la Directory debba essere

riorganizzata.
Infatti quando dati e programmi sono stati ripetutamente salvati (SAVE) e cancellati (SCRATCH), di queste operazioni possono esserci rimaste numerose tracce, in particolare in piccoli blocchi sparpagliati, appunto troppo piccoli perche' possano essere riutilizzati.
In effetti la funzione di VALIDATE e' quella di ricostruire la BAM del disco leggendo i file presenti nella DIRECTORY.
Se durante questa operazione viene incontrato un errore allora la funzione di ricostruzione viene sospesa e si ritorna alle condizioni di partenza.

Il comando VALIDATE riorganizzerra' allora il dischetto in modo tale che si possa disporre del massimo spazio effettivamente disponibile.

!!!ATTENZIONE!!!

C' e' un pericolo nell' uso di questo comando. Quando si usino i FILES RANDOM i blocchi ALLOCATI saranno DE-ALLOCATI con questo comando. Per questo motivo il VALIDATE non dovrebbe mai essere usato quando in un dischetto sono presenti i files random.

Naturalmente e' un comando che si usa in forma diretta in una delle due forme:

```
PRINT#15, "VALIDATE"

PRINT#15,"V"
```

RENAME

Questo comando consente di cambiare nome ad un file di programmi o di dati.
In effetti si tratta di una operazione molto veloce perche' l' unico cambiamento che avviene e' nella Directory del disco.
Sul disco naturalmente non deve esistere gia' un file con lo stesso nome utilizzato nel RENAME perche' in questo caso l' operazione non potra' avvenire ed avremo una segnalazione di errore:

FILE EXISTS.

Il formato di RENAME e':

```
PRINT#15,"RENAME0:vecchio nome=nuovo nome"
```

o nella forma abbreviata R al posto della lettera RENAME.

SCRATCH

Questo comando consente di cancellare files e programmi dal disco rendendo disponibili i blocchi per nuove informazioni.
Si possono cancellare programmi uno alla volta o in gruppo come possiamo vedere dagli esempi. Il formato generale del programma e' il seguente:

PRINT#15,"SCRATCH0:nome del programma"

o abbreviando, S al posto della parola SCRATCH.

ESEMPI

Ammettiamo che siano presenti i seguenti files:

TEST,TRAIN,TRUCK,TAIL

possiamo usare:

PRINT#15,"S0:TR*"

se si desideri cancellare sia TRAIN che TRUCK. Usando invece:

PRINT#15,"S0:T*"

li cancelleremo tutti. Cancelleremo cioe' tutti i files che iniziano per T.

Se per esempio la directory contenesse i files KNOW e GNAW usando:

PRINT#15,"S0:?N?W"

cancelleremo ambedue i programmi in quanto il ? sostituisce i caratteri ignoti all' inizio o nel mezzo del nome del file.

COPY

Questo comando consente, come del resto e' implicito nel nome, di effettuare una copia di un qualsiasi file di dati o programmi.
Nel caso si disponga di un solo drive e' ovvio che la copia puo' essere fatta solo sullo stesso dischetto.
Il formato di questo comando e':

```
PRINT#15,"COPY0:nuovo file=0:vecchio file"
```

oppure usando la lettera C al posto della parola COPY.

Questo comando puo' anche essere usato in modo interessante per la concatenazione di files sequenziali.

Esempio:

```
PRINT#15,"C0:MAILING FILE=0:NOME,0:INDIRIZZO,0:TELEFONO"
```

CAPITOLO TERZO

CARICAMENTO DI UN PROGRAMMA IN LINGUAGGIO MACCHINA

I programmi in linguaggio macchina sono manipolati in maniera leggermente differrente rispetto ai programmi Basic.
Un programma in linguaggio macchina e' trasferito dal floppy al computer utilizzando un indirizzo secondario uguale a 1.Quando viene utilizzato un indirizzo secondario uguale a 1 il programma e' caricato in modo assoluto il che equivale a dire che viene caricato nella memoria incominciando dall'indirizzo specificato nei primi 2 byte del file su disco e non come i normali programmi Basic a partire dall'inizio della memoria disponibile. Un esempio:

```
LOAD "PROGRAMMA MACCHINA",8,1
```

Carichera' il programma in linguaggio macchina a un indirizzo assoluto. Per esempio il programma puo' essere stato fissato perche' parta dall'indirizzo decimale49152 ed andra' pertanto in esecuzione con un comando:

```
SYS 49152.
```

Se provate a caricare un programma in linguaggio macchine senza l'indirizzo secondario e provate a digitare il RUN molto probabilmente vedrete apparire il messaggio:

```
SYNTAX ERROR IN......
```

Allo stesso modo, provando ad eseguire il LIST del

programma in linguaggio macchina avrete visualizzato dei dati che non hanno alcun senso.
E' da notare che i programmi in linguaggio macchina non hanno una differrenziazione particolare dai programmi in basic nella directory, infatti entrambi vengono visualizzati, memorizzati come files di tipo PRG cioe' PROGRAM.
Normalmente possiamo concludere che se dopo aver scritto RUN risultera' un messaggio di:

SYNTAX ERROR IN....

sara' un programma non scritto in Basic ma in linguaggio macchina. In questo caso dovremo resettare il computer anche con un SYS 63478 e ricaricare con il comando:

LOAD"NOME DEL PROGRAMMA",8,1.

Tuttavia il programma potrebbe non partire con un RUN, in quanto potrebbe avere la necessita' di un SYS cioe' di un salto a quel determinato indirizzo. Per questo sara' necessario trovare l'indirizzo di partenza di questo programma. Successivamente spiegheremo portando anche un esempio come fare a listare tutti i parametri relativi a un programma.
Uno di questi parametri e'l'indirizzo di caricamento. Questo indirizzo di caricamento o LOAD ADDRESS e' normalmente l'indirizzo di esecuzione iniziale del programma e puo' essere richiamato con um comando:

SYS LOAD ADDRESS.

Si puo' trovare l'indirizzo iniziale o il LOAD ADDRESS di un programma con la seguente utility:

```
10 OPEN 1,8,2,"nome programma,P,R"
```

```
20 GET#1,X$:IF X$="" THEN X$=CHR$(0)
30 LB=ASC(X$)
40 GET#1,X$:IF X$="" THEN X$=CHR$(0)
50 HB=ASC(X$)
60 CLOSE 1
70 AD=HB*256+LB
80 PRINT"LOAD ADRESS:";AD
```

Il programma mostrera' il LOAD ADRESS di un programma qualsiasi. Qui il file programma e' aperto come un file di dati. L'indirizzo di partenza e' immagazzinato come i primi due byte del file e letto utlilizzando un comando GET e un'appropriata tecnica di programmazione. Il primo byte e' il byte di ordine basso mentre il secondo byte e' il byte di ordine alto che insieme compongono i due byte dell'indirizzo.

IMMAGAZZINAMENTO DI PROGRAMMI IN LINGUAGGIO MACCHINA

I programmi in linguaggio macchina sono normalmente scritti con un assemblatore oppure con un monitor e salvati utilizzando le funzioni dell'uno o dell'altro. Tuttavia ricordiamo che i programmi in linguaggio macchina possono essere anche scritti attraverso una piccola routine di basic con i byte individuali del programma scritti come valori decimali e immessi in dei comandi DATA. Vediamo un esempio di programma in linguaggio macchina scritto in basic con l'aggiunta di comandi DATA:

```
10 SA=indirizzo di inizio
20 EA=indirizzo di fine
30 FOR I=SA TO EA
```

```
40 READ X
50 POKE I, X
60 NEXT I
80 DATA ..............................
90 DATA ..............................
```

In questo esempio il valore decimale dell'indirizzo di partenza e' nella linea 10 mentre il valore di fine e' nella linea 20.
I valori decimali dei singoli byte del programma in linguaggio macchina sono inseriti nei comandi DATA nella parte finale del programma stesso separati da virgola. Con questo sistema possono essere quindi scritti utilizzando il basic dei programmi in linguaggio macchina. Tuttavia questo sistema e' estramamente lungo e complicato.
Un metodo migliore e che comunque fa risparmiare tempo e' di immagazzinare i programmi in linguaggio macchina nella loro forma reale. Con questo sistema potrete immediatamente eseguire il programma dopo averlo caricato (LOAD) senza che si renda necessario una conversione che come abbiamo visto e' abbastanza complicato. Utilizzando il seguente esempio potremo salvare un programma che e' gia' in memoria:

```
10 SA=indirizzo di inizio
20 EA=indirizzo di fine
30 OPEN 1,8,1,"0:nome programma,p,w"
40 HB=INT(SA/256):LB=SA-HB*256
50 PRINT#1,CHR$(LB);CHR$(HB);
60 FOR I=SA TO EA
70 PRINT#1,CHR$(PEEK(I));
80 NEXT I
90 CLOSE 1
```

Come abbiamo detto questa routine parte dal concetto che il programma in linguaggio macchina sia gia' nella memoria del computer. Se un programma e' gia' codificato o inserito in comandi DATA la seguente routine puo' essere utilizzata per produrre un programma in puro linguaggio macchina:

```
10 SA=indirizzo di inizio
20 EA=indirizzo di fine
30 OPEN 1,8,1"0:nome programma,p,w"
40 HB=INT(SA/256):LB=SA-HB*256
50 PRINT#1,CHR$(LB);CHR$(HB);
60 FOR I=SA TO EA
70 READ X
80 PRINT#1,CHR$(X);
90 NEXT I
100 CLOSE 1
110 DATA............................
120 DATA............................
```

La precedente routine scrive un programma in linguaggio macchina su dischetto che potra' essere piu' tardi ricaricato con un comando:

LOAD"nome del programma",8,1.

Quindi il programma stesso potra'essere eseguito con un comando:

SYS(indirizzo d'inizio)

Programmi in linguaggio macchina possono anche essere caricati ed eseguiti da programmi basic. Un programma di questo genere deve avere la seguentte forma:

```
10 IF A=0 THEN A=1:LOAD"nome programma",8,1
20 SYS (starting address)
```

Il comando IF nella linea 10 puo' generare confusione. Tuttavia esso deve essere presente perche' dopo l'esecuzione di un LOAD all'interno di un programma l'interpretre basic incomincerebbe l'esecuzine sempre alla prima linea del programma basic. Poiche' il programma in linguaggio macchina normalmente non si sovrappone al' programmma basic in memoria, il programma basic originale rimane intatto e viene rieseguito. Se usate la routine:

```
10 LOAD"nome programma",8,1
20 SYS (indirizzo di inizio)
```

Il programma continua a ricaricare, cioe' a eseguire un LOAD ancora una volta" sul nome del programma" e il comando SYS non viene mai eseguito. Se la variabile A e' presente il programma salta alla linea 20 al termine del primo comando presente sulla linea 10. Questo esempio di caricamento chiamato anche "LOADER" puo' essere messo sul dischetto insieme al programma in linguaggio macchina. Per eseguire il programmma in linguaggio macchina sara' necessario dare solamente il comando:

```
LOAD"loader",8
RUN
```

Questo sistema ha il vantaggio che l'indirizzo d'inizio del programmma in linguaggio macchina non e' necessario sia conosciuto perche' e' incluso nell'indirizzo SYS del LOADER.

CAPITOLO QUARTO

TRATTAMENTO DI FILES SEQUENZIALI COME TAVOLE

Affinche' i dati possano essere correttamente manipolati un file di dati sequenziali deve risiedere completamente nella memoria del computer. Il piu' delle volte puo' essere usata una tavola a due dimensioni. Questa tavola e' anche chiamata matrice perche' ogni suo elemento puo' essere indirizzato attraverso due coordinate. A questo fine potrete usare una variabile a due dimensioni che dovra' essere riservata con un comando DIM. Il primo indice corrispondera' al data record, il secondo indice di dimensionamento al campo entro il record. Il seguente diagramma mostra un esempio di una tavola:

	Field 1	Field 2	Field 3
Record1	D$(1,1)	D$(1,2)	D$(1,3)
Record2	D$(2,1)	D$(2,2)	D$(2,3)
Record3	D$(3,1)	D$(3,2)	D$(3,3)
Record4	D$(4,1)	D$(4,2)	D$(4,3)
Record5	D$(5,1)	D$(5,2)	D$(5,3)
Record6	D$(6,1)	D$(6,2)	D$(6,3)

Questa tavola e' un file completo di sei records che hanno tre campi ciascuno. La variabile D$ e' riservata con un comando:

DIM D$(6,3).

Per leggere un file sequenziale come una tavola e' necessario di creare un tale file con, per esempio, sei records con ,tre campi ciascuno. Per far questo utilizzare il seguente programma:

```
100 OPEN 1,8,2"TABFILE,S,W"
110 FOR X=1 TO 6
120 PRINT CHR$(147)
130 PRINT"RECORD ";X
140 PRINT"---------"
150 FOR Y=1 TO 3
160 PRINT"CAMPO ";Y;": ";
170 INPUT X$
180 PRINT#1,X$
190 NEXT Y
200 NEXT X
210 CLOSE 1
```

In questo programmma sono utilizzati due cicli le cui variabili sono numerate con record e file. I cicli incominciano rispettivamente alla linea 110 e alla linea 150. Inserire quindi sei records. Quando il programma e' stato eseguito questi records saranno contenuti nel dischetto con il nome di TABFILE.
Ricordarsi di salvare questo programma con il comando:

SAVE"TABPROG",8

in maniera tale che possiate usarlo successivamente.

Questo file puo' essere ora caricato dentro il computer come una tavola o matrice. Sono necessari due cicli indicizzàti per la tavola:

```
100 OPEN 1,8,2,"TABFILE.SEQ,S,R"
110 DIM D$(6,3)
120 FOR X=1 TO 6
130 FOR Y=1 TO 3
140 INPUT#1,D$(X,Y)
150 NEXT Y
160 NEXT X
170 CLOSE 1
```

Questo programma immette i dati entro la tavola. Potete controllare cio' con una serie di comandi PRINT per vedere se i dati sono stati inseriti nel giusto posto. Poiche' ogni campo puo' essere indirizzato con indici, potete dare un comando come:

```
PRINT D$(1,2)
```

per esaminare il secondo campo del primo record. E' di grande importanza essere capaci di visualizzare i campi di un dato record. Per questo utilizzare la seguente routine ricordandosi di aver prima pero' salvato il precedente programma:

```
100 INPUT"NUMERO RECORD: ";X
110 PRINT"-------------------"
120 PRINT"CAMPO 1: ";D$(X,1)
130 PRINT"CAMPO 2: ";D$(X,2)
140 PRINT"CAMPO 3: ";D$(X,3)
```

La tavola che abbiamo visto puo' essere cambiata come si vuole. Aggiungere per esempio le seguenti righe al precedente programma:

```
160 PRINT"------------------"
170 INPUT"CAMPO DA CAMBIARE:";Y
180 INPUT"NUOVI CONTENUTI:   ";D$(X,Y)
190 PRINT"OK
200 PRINT"ALTRI CAMBIAMENTI (S/N)?"
210 GET X$:IF X$="" THEN 210
220 IF X$="S" THEN 100
230 IF X$="N" THEN END
240 GOTO 210
```

In questo caso il numero del campo che deve essere cambiato e' utilizzato come secondo indice. Questi e' accanto all'indice del record richiesto per inserire un nuovo elemento nella tavola.
Questa tavola modificata deve ora essere scritta ancora una volta su dischetto. Per poter utilizzare la seguente routine, ricordando di non dimenticare di eseguire un SAVE del precedente programma.

```
100 OPEN 1,8,2,"@:TABFILE,S,W"
110 FOR X=1 TO 6
120 FOR Y=1 TO 3
130 PRINT#1, D$(X,Y)
140 NEXT Y
150 NEXT X
160 CLOSE 1
```

Anche questa routine e' relativamente corta a causa dell'uso dei cicli nidificatti.
Il simbolo @: nella linea 10 e' necessario per sovrascrivere il file esistente, cioe' per scriverci sopra.

L'accesso ai dati attraverso l'impiego di una tavola e' molto veloce. I tempi di accesso sono indipendenti dalla grandezza della tavola. Le dimensioni della tavola e quindi la quantita' di dati che si possono manipolare dipende dalla capacita' di memoria del computer, tuttavia il COMMODORE 64 che e' dotato di una grande dimensione di memoria si presta a meraviglia per questi impieghi. Infatti se scrivete un programma di manipolazione dati che occupa 8K bytes di memoria vi rimarranno ancora 30K bytes per immagazzinare dati. Se considerate quindi che l'immagazzinamento di un record contenente un nome e un indirizzo richiede circa 80 caratteri potrete notare che vi rimane spazio per immagazzinare ben 384 records in memoria. E questo con un tempo di accesso che non puo' essere superato da qualsiasi altro sistema di manipolazione dati o da qualsiasi altra tecnica. Tuttavia con grande quantita' di dati, l'immagazzinamento con sistemi sequenziali non e' impiegabile.

TAVOLE DI RICERCA

Come abbiamo detto in precedenza ogni record di dati di una tavola puo' essere indicizzato. Poiche' si tratta di una tavola a due dimensioni il primo indice seleziona il record. Per questo, se si deve accedere a un record della tavola, o se questo comunque debba essere cambiato, l'operatore deve conoscere il numero del record. Tuttavia ci sono dèi files di dati per i quali questo metodo non e' conveniente, in altre parole non esiste un sistema semplice e di facile applicazione nelle numerazione dei records stessi.
In questi files il numero del record deve essere trovato

attraverso una ricerca fra tutti i records. Ecco un esempio pratico di quanto abbiamo detto.
Per prima cosa create un file di dati con il seguente programma. Nomi e numeri di telefono vengono salvati con il seguente esempio:

```
100 OPEN 1,8,2,"TELEDAT,S,W"
110 PRINT CHR$(147)
120 INPUT"COGNOME      :";LN$
130 INPUT"NOME         :";FN$
140 INPUT"C.A.P.       :";AC$
150 INPUT"TELEFONO     :";NU$
160 PRINT"INFORMAZIONE CORRETTA (S/N)?"
170 GETX$:IF X$="" OR X$<>"S"
    AND X$<>"N" THEN 170
180 IF X$="N" THEN 110
190 PRINT#1,LN$","FN$","AC$","NU$
200 PRINT"ALTRI DATI (S/N)?"
210 GETX$:IF X$="" OR X$<>"S"
    AND X$<>"N" THEN 210
220 IF X$="N" THEN 240
230 GOTO 110
240 CLOSE 1
```

COMMENTO AL PROGRAMMA

Linea 100
Il file sequenziale "TELEDAT" e' aperto per la scrittura.
Linea 110
Lo schermo viene ripulito.
Linee da 120-150
Sono inseriti i quattro campi da tastiera.
Linee da 160-180
Se i dati inseriti non sono corretti successivamente possono essere ancora cambiati.

Linea 190
I quattro campi sono scritti su disco.
Linee da 200-220
Qui l'esecuzione del programma puo' essere terminata.
Linea 230
Possiamo continuare l'input.
Linea 240
Il file aperto nella linea 100 viene chiuso.

Scrivete questo programma, fatelo girare e inserite alcuni dati. Ricordatevi di salvare il programma sul dischetto in modo tale da poterlo successivamente combinare con altri sottoprogrammi o routines.
Se avete inserito alcuni dati probabilmente vorrete trovare un numero di telefono. Per far cio' dovreste stampare l'intero file sulla stampante, o visualizzarlo sullo schermo e trovarvelo da Voi. Tuttavia questo e' un metodo abbastanza lento specialmente se si sono inseriti molti records.
La ricerca per un numero di telefono corrispondente a un dato nome puo' essere eseguita dal computer. Il sistema dovrebbe cercare nell'intera lista, trovare il nome desiderato e una volta trovato restituirvi il record completo che contiene quel nome. La seguente routine vi mostra come fare:

```
100 OPEN 1,8,2,"TELEDAT,S,R"
110 DIM D$(100,4):X=1
120 INPUT#1,D$(X,1),D$(X,2),D$(X,3),
    D$(X,4)
130 IF ST<>64 THEN X=X+1:GOTO 120
140 CLOSE 1
150 PRINT CHR$(147)
160 PRINT"COGNOME??? :   ";N$
```

```
170 FOR I=1 TO X
180 IF D$(I,1)=N$ THEN 210
190 NEXT I
200 PRINT"COGNOME NON TROVATO!":GOTO 280
210 PRINT"COGNOME TROVATO:"
220 PRINT"-----------"
230 PRINT"COGNOME  :  ";D$(I,1)
240 PRINT"NOME      : ";D$(I,2)
250 PRINT"C.A.P.   :  ";D$(I,3)
260 PRINT"TELEFONO:   ";D$(I,4)
270 PRINT"-----------"
280 PRINT"ALTRO (S/N)?"
290 GETX$:IF X$="" OR X$<>"S"AND X$<>"N" THEN 290
300 IF X$="S" THEN 150
310 PRINT"FINE PROGRAMMA":END
```

COMMENTO AL PROGRAMMA

Linea 100
Il file sequenziale "TELEDAT" e' aperto per la lettura.
Linea 110-120
La tavola e' dimensionata per 100 records e l'indice viene letto entro la tavola
Linea 130
La variabile di stato ST e' controllata per vederre se siamo alla fine di un file che verrebbe indicato con un valore di 64. Se non siamo alla fine, cioe' se ST non e' uguale a 64 l'indice e'incrementato e viene letto un nuovo record.
Linea 140
Il file aperto nella linea 100 e' chiuso.
Linea 150
Lo schermo viene pulito.
Linea 160
L'ultimo nome che deve essere cercato e' letto da tastiera e immesso nella variabile N$.

Linee da 170-190
Il ciclo ricerca la tavola dei records, confrontando il nome del campo con il nome desiderato. Se viene trovata la corretta posizione il programma salta alla routine diuscita.
Linea 200
Il nome non e' stato trovato.
Linee da 210-270
Il record contenente il nome desiderato viene visualizzato.
Linee da 280-310
Viene consentita la possibilita' di ricercare il nuovo nome.

Potete notare che questa tecnica di ricerca e' abbastanza veloce quando i dati sono gia' caricati nella memoria interna del computer. Infatti la ricerca all'interno della memoria del computer e' piu' veloce che la ricerca su dischetto.
Il programma appena visto puo' anche essere facilmente cambiato per ricercare un altro campo invece del nome. Potete ricercare per il codice di avviamento postale per esempio. Il primo programma arresta la ricerca quando il primo record di dati che si voleva e' trovato. Cio' tuttavia puo' essere non sempre valido. Infatti per esempio si potrebbe voler ricercare attraverso tutta la tavola per un particolare codice di avviamento postale e desiderare che tutti quei codici di avviamento postale siano visualizzati, per questo sara' necessaria una routine diversa.Infatti questa routine deve continuare la ricerca anche dopo che il primo oggetto della nostra ricerca sia stato trovato. Il programma che vi presentiamo esegue quanto richiesto:

```
100 OPEN 1,8,2,"TELEDAT,S,R"
110 DIM D$(100,4):X=1
120 INPUT#1,D$(X,1),D$(X,2),D$(X,3),
    D$(X,4)
130 IF ST<>64 THEN X=X+1:GOTO 120
140 CLOSE 1
150 PRINT CHR$(147)
160 PRINT"RICERCA PER C.A.P.: ";AC$
170 FOR I=1 TO X
180 IF D$(I,3)=AC$ THEN 210
190 NEXT I
200 PRINT"FINE DATI!":GOTO 310
210 PRINT"----------------"
220 PRINT"COGNOME:        ";D$(I,1)
230 PRINT"NOME    :       ";D$(I,2)
240 PRINT"C.A.P. :        ";D$(I,3)
250 PRINT"TELEFONO:       ";D$(I,4)
260 PRINT"----------------"
270 PRINT"ALTRO (S/N)?"
280 GETX$:IF X$="" OR X$<>"S"
    AND X$<>"N" THEN 280
290 IF X$="S" THEN 190
300 PRINT"RICERCA ESEGUITA"
310 PRINT"ALTRA RRICERCA (S/N)?"
320 GETX$:IFX$$="" OR X$<>"S"
    AND X$<>"N"THEN320
330 IF X$="S"THEN150
340 END
```

Qui la ricerca continua se un record con il giusto codice di avviamento postale e' trovato. Questo succede nella linea 290 che ritorna indietro al ciclo di ricerca invece di far terminare l'esecuzione del programma stesso. Dopo la ricerca attraverso tutti i records il

programma risponde con un:

FINE DATI

Se avete ben compreso le operazioni che sono state eseguite fino ad ora potrete a questo punto scrivere un programma con ricerche diverse senza alcuna difficolta'.

I SORT

Nell'elaborazione dei dati e' spesso necessario eseguire un ordinamento numerico o alfabetico.
Possiamo dire che questo fatto si e' verificato fino dall'inserimento nel primo computer e i programmatori hanno sempre cercato di risparmiare tempo e lavoro sviluppando sempre migliori metodi di ordinamento o SORT. Dobbiamo inoltre dire che il SORT consuma del tempo, in maniera particolare quando viene eseguito con un linguaggio di per se lento come il basic. Ma vediamo il perche' del sort. Supponiamo di avere un elenco telefonico in cui i nomi non sono in ordine. Malgrado la velocita' del disco dovrete ugualmente ricercare,probabilmente sull'intero elenco, dall'inizio alla fine per ritrovare il nome.
Le tecniche di SORT offrono il vantaggio di diminuire in maniera enorme il tempo necessario alla ricerca dei dati. Esistono diverse tecniche di ricerca, o metodi, che differiscono principalmente nella velocita' di esecuzione. Il metodo piu' semplice confronta ogni dato con gli altri. Se si suppone che una tavola debba essere messa in ordine ascendente il primo campo della tavola sara' confrontato con il secondo. Se il primo e' piu' grande sara' scambiato con il secondo. Dopo di cio' il

primo sara' confrontato con il terzo e cosi' via fino a quando l'ultimo dato non verra' confrontato e messo nel giusto ordine. A questo punto il campo piu' piccolo sara' all'inizio del file nel giusto ordine.
Questo metodo potrebbe sembrare un tantino complicato, tuttavia eseguito in memoria e' abbastanza veloce. Dobbiamo ricordare pero' che questo sistema e' sufficiente per piccole quantita' di dati.
Per far girare questo programma, cioe' per definire questo esempio per prima cosa deve essere costruita una tavola. Il seguente esempio utilizza una tavola con dodici voci contenenti dati alfanumerici e stringhe. La tavola viene riempita dalla seguente routine:

```
100 DIM TA$(12)
110 FOR I=1 TO 12
120 INPUT TA$(I)
130 NEXT I
```

Questo programma consente di inserire 12 stringhe sulle quali sara' quindi eseguito un SORT con il seguente programma:

```
140 I=1
150 X=I+1
160 IF TA$(I) < TA$(X) THEN 180
170 TA$(0)=TA$(I):TA$(I)=TA$(X):
    TA$(X)=TA$(0)
180 X=X+1
190 IF X<= 12 THEN 160
200 I=I+1
210 IF I <>12 THEN 150
220 FOR I=1 TO 12
```

```
230 PRINT TA$(12)
240 NEXT I
```

La tavola e' ordinata a visualizzata sullo schermo. Se invece di una tavola ad una sola dimensione si desidera eseguire un SORT su tavole a due dimensioni, come il nostro file di numeri telefonici, occorrera' cambiare dei campi eseguendo delle variazioni alle linee 160-170 come segue:

```
160 IF D$(I,1) < D$(X,1) THEN 180
170 D$(0,1)=D$(I,1):D$(I,1)=D$(X,1);
    D$(X,1)=D$(0,1)
171 D$(0,2)=D$(I,2):D$(I,2)=D$(X,2):
    D$(X,2)=D$(0,2)
172 D$(0,3)=D$(I,3):D$(I,3)=D$(X,3):
    D$(X,3)=D$(0,3)
173 D$(0,4)=D4(I,4):D$(I,4)=D$(X,4):
    D$(X,4)=D4(0,4)
```

NOTA FINALE

Naturalmente come abbiamo detto all'inizio questi sono programmi scritti in basic per cui abbastanza lenti. Se volete utilizzare dei programmi piu' veloci dovrete impiegare l'assembler per il quale vi consigliamo "IL CORSO DI ASSEMBLER II"edito da E.V.M.

CAPITOLO QUINTO

LA STRUTTURA DEL DISCHETTO 1541

Il dischetto del 1541 e' diviso in 35 tracce. Ogni traccia contiene un numero variabile da 17 a 21 settori. Il numero totale di settori e' 683. Poiche' la directory occupa la traccia 18 sono in effetti disponibili 664 settori, ognuno dei quali contiene 256 bytes. La seguente tabella mostra la distribuzione delle tracce e dei settori per traccia:

TRACCIA	NUMERO DEL SETTORE
1 A 17	21
18 A 24	19
25 A 30	18
31 A 35	17

LA BAM DEL 1541

BAM e' un'abbreviazione di Block Availability Map, cioe' mappa di disponibilita' dei blocchi. In essa infatti la BAM indica dove, sul dischetto, un blocco e' libero o occupato da un file. Dopo una qualsiasi operazione su blocchi (operazione di SAVE, di DELETE, eccetera) la BAM viene aggiornata. Quando la BAM indica che un file che deve essere salvato richiede piu' blocchi di quanti ce ne siano disponibili verra' segnalato un messaggio di errore. Quando un file e' aperto la BAM nel DOS e'

aggiornata e riscritta su disco quando il file viene chiuso. Quindi comandi che hanno funzioni di scrittura o di cancellazione leggono la BAM, l'aggiornano e la riscrivono quindi sul dischetto.
E' facile capire che invece i comandi di lettura non eseguono nessun aggiornamento tanto e' vero che possono essere letti i dati anche da un dischetto protetto. La BAM sulla traccia 18 settore 0 e' organizzata nella seguente maniera:

TRACCIA 18, SETTORE 0

BYTE		CONTENUTO	NOTE
0,1	($00-$01)	$12,$01	TRA.,SETT.DEL PRIMO BLOCCO DELLA DIRECTORY
2	($02)	$41	ASCII DI 'A'
3	($03)	$00	USI FUTURI
4-143	($04-$8F)		*BIT MAP DEI BLOCCHI USATI E LIBERI

* 1= blocchi liberi; 0 = blocchi usati

Il bit map dei blocchi e' organizzato in maniera tale che quattro bytes rappresentino il settore sulla traccia. Come puo' essere ben compreso dalla seguente tavola il primo dei quattro bytes contiene il numero dei blocchi liberi sulla traccia. Gli altri tre bytes (24 bits) indicano quali blocchi sono liberi e quali sono gia' stati utilizzati in quella traccia.

STRUTTURA DELLA BAM PER OGNI TRACCIA

BYTE	CONTENUTO
0	Numero dei blocchi disponibili nella traccia.
1	Bit map dei settori 0-7
2	Bit map dei settori 8-15
3	Bit map dei settori 16-23

ESEMPIO DEL CONTENUTO DI 4 BYTES DI UNA TRACCIA NELLA BAM

TRACCIA 18,SETT. 0, BYTES 4-7 (TRACK1)

```
00001010  00000000 00000011 11111111
 ($0A)     ($00)     ($03)    ($FF)

10 FREE    1 = LIBERI
BLOCKS     0 = UTILIZZATI
```

Utilizzando un semplice programma potrete leggere il primo byte di ogni traccia nella bit map che ci da il numero di settori liberi in quella traccia , sommarli insieme e trovare il numero totale di blocchi liberi sul dischetto.

Esempio:

```
10 OPEN15,8,15,"I0"
20 OPEN2,8,2,"#"
30 PRINT#15,"U1";2;0;18;0
40 FORI=1TO35:IFI=18THENNEXT
50 PRINT#15,"B-P";2;(I*4):GET#2,X$:IFX$=""THENX$=CHR$(0)
```

```
50 TB=TB+ASC(X$):NEXT
60 CLOSE2:CLOSE15:?"BLOCCHI LIBERI "TB
```

LA DIRECTORY

La directory e' la tavola di cio' che contiene il dischetto. Contiene le seguenti informazioni:

-nome del disco
-identificatore del disco (ID)
-numero di versione del sistema operativ o su disco (DOS)
-nomi dei file
-tipi dei file
-blocchi per ogni file
-blocchi liberi

Questa directory e' caricata nella memoria dell'unita' centrale con il comando:
LOAD"$",8.
Attenzione perche' un programmma preventivamente caricato in memoria sara' distrutto.
Puo' essere visualizzata una volta caricata con un comando LIST.
La directory occupa tutta la traccia 18 del dischetto. La lista dei files segue la testata della directory o DIRECTORY HEADER. Ogni blocco puo' contenere una lista massima di 8 files. Poiche' la diciottesima traccia ha 19 blocchi e poiche' la Bam e la DIRECTORY HEADER occupano un blocco solo 18 blocchi rimangono per la lista dei files. Per questo motivo sul dischetto possono risiedere un massimo di 144 files che e' dato da 18 blocchi per 8 punti d'ingresso ciascuno.

FORMATO DEL DIRECTORY HEADER

TRACCIA 18, SETTORE 0

BYTE		CONTENUTI	NOTE
144-161	($90-$A1)		A)
162,163	($A2-$A3)		B)
164	($A4)	$A0	C)
165,166	($A5-$A6)	$32,$41	D)
167-170	($A7-$AA)	$A0	E)
171-225	($AB-$FF)	$00	NON USATI

A) Nome del disco.Se inferiore a 16 caratteri riempito con spazi shiftati.

B) Identificatore disco o ID

C) Spazio shiftato.

D) Carattere ASCII "2A" che sta ad indicare il formato di questa unita' a dischi.

E) Come punto C)

IL NOME DEL DISCHETTO

Il nome del dischetto puo' essere di un massimo di 16 caratteri di lunghezza e viene stabilito all'atto della formattazione del dischetto stesso.
Se viene assegnato un nome inferiore a 16 caratteri il resto dello spazio disponibile viene riempito con spazi shiftati ($A0). La seguente routine basic legge il nome, e lo salva, in una stringa DN$:

```
100 OPEN 15,8,15,"IO"
101 REM APERTURA DEL CANALE DI COMANDO E INIZIALIZZAZIONE
110 OPEN 2,8,2,"#"
111 REM APERTURA CANALE 2
120 PRINT#15,"B-R";2;0;18;0
121 REM TRACCIA 18 E SETTORE 0 LETTI ED IMMESSI NEL CANALE 2
130 PRINT#15,"B-P";2;144
131 REM BUFFER-POINTER AL BYTE 144
140 DN$=""
141 REM LA STRINGA DN$ E' CANCELLATA
150 REM CICLO DI LETTURA
160 FOR I=1 TO 16
170 GET#2,X$
171 REM LEGGE UN BYTE
180 IF ASC(X$)=160 THEN200
181 REM IGNORA GLI SPAZI SHIFT
190 DN$=DN$+X$
191 REM BYTE AGGIUNTO A DN$
200 NEXT 1
210 CLOSE 2:CLOSE 15
211 REM CHIUSURA DEI CANALI
```

Dopo aver fatto girare questa routine la stringa DN$

contiene il nome del disco

IDENTIFICATORI DEL DISCHETTO

L'identificatore del dischetto (ID) e' di 2 caratteri di lunghezza e viene specificato anche questo all'atto della formattazione del dischetto.
Il DOS usa questo identificatore per controllare se il dischetto presente in quel momento nel drive e' stato sostituito. Se e' cosi' allora il DOS esegue un processo di inizializzazione. Inizializzando un dischetto si carichera' la BAM entro la memoria del drive. Con questo sistema la BAM sulla quale si deve operare e' sempre in memoria, naturalmente ricordandosi di formattare sempre con differenti identificatori. E' da notare che se il sistema operativo non esegue questa funzione appena decritta questa dovrebbe essere fatta utilizzando un comando d'inizializzazione.

IL FORMATO DELLA DIRECTORY

In un File Entry sono presenti i parametri relativi a l' indirizzo TRACCIA e SETTORE del file e i parametri che lo definiscono.
I settori da 1 a 19 sulla traccia 18 contengono i files entry. I primi due bytes di un settore puntano al successivo settore della directory . In altre parole ne contengono gli indirizzi.
Se non ci sono altri blocchi nella directory allora questi bytes conterranno i valori $00 e $FF rispettivamente.

TRACCIA 18, SETTORE 1

BYTE	CONTENUTI			
0,1	($00,$01)	N.TR E SETT. BLOCCO SUCCESS.		
2-31	($02-$1F)	ENTRY	1	FILE
34-63	($22-$3F)	"	2	"
66-95	($42-$5F)	"	3	"
98-127	($62-$7F)	"	4	"
130-159	($82-$9F)	"	5	"
162-191	($A2-$BF)	"	6	"
194-223	($C2-$DF)	"	7	"
226-255	($E2-$FF)	"	8	"

FORMATO DEGLI ENTRY FILE DELLA DIRECTORY

Ogni ENTRY FILE consiste in 30 bytes le cui funzioni sono descritte di seguito

BYTE	CONTENUTI	
0	($00)	TIPO FILE
1,2	($01,$02)	N.TRAC. E SETT.PRIMO BLOCCO DATI
3-18	($03-$12)	NOME FILE
19,20	($13,$14)	PER RELATIVES (SIDE-SECTOR)*
21	($15)	C.S.(LUNGH. RECORD)
22-25	($16-$19)	NON USATI
26,27	($1A-$1B)	**
28,29	($1C-$1D)	N.BLOCCHI DEL FILE

*Come vedremo a proposito dei Relatives, questi Bytes contengono l' indirizzo di traccia e settore del primo blocco di SIDE-SECTOR
**Numero di traccia e settore del nuovo file quando viene riscritto utilizzando un comando ò.

SEGNALATORI DI TIPO FILE

Il byte 0 dell' ENTRY file indica su che tipo di file stiamo lavorando. I bits da 0 a 2 sono utilizzati per indicare 5 tipi di files.
Il bit 7 indica se il file e' stato chiuso appropriatamente. La corretta chiusura di un file mette a uno il bit 7. Un file non chiuso viene segnalato da un asterisco che precede il tipo del file nel listato della directory. Se per esempio viene aperto un file di nome "TEST" senza richiuderlo e viene listata la directory questo file verra' rappresentato in questo modo:

"TEST"   *SEQ

Non appena il file verra' chiuso l'asterisco non apparira' piu' nel listato della directory. Se questo file rimane aperto e successivamente si tenta di riaprirlo di nuovo verra' visualizzato il messaggio di errore:

"WRITE FILE OPEN"

TIPI DI FILE

Per comprendere esattamente la funzione del byte 0 nell' ENTRY FILE, il tipo del file, vediamo ora una tavola di tutti i tipi di file:

File type	Bit mask opened		Bit mask closed	
	7654 3210	hex	7654 3210	hex
DELeted	0000 0000	$00	1000 0000	$80
SEQuential	0000 0001	$01	1000 0001	$81
ProGram	0000 0010	$02	1000 0010	$82
USeR	0000 0011	$03	1000 0011	$83
RELative	0000 0100	$04	1000 0100	$84

Avrete notato che i bits da 3 a 6 sembra non abbiano alcuna funzione. Ma abbiamo verificato con l'aiuto del listato del sistema operativo che il bit 6 ha invece una funzione. Questa e' che il bit 6 di un file denota un file protetto. Se avete messo a uno questo bit il corrispondente file non potra' essere piu' cancellato. Consultare il listato del disassemblato per maggiori informazioni.

TRACCIA E SETTORE DEL PRIMO BLOCCO DATI

I bytes 1 e 2 dell' FILE ENTRY indirizzano cioe' puntano al primo blocco dati del file. Il primo byte contiene la traccia e il secondo il numero di settore dove il file inizia. Il primo blocco dati contiene un puntatore al secondo blocco del file che e' anche contenuto nei primi due bytes del blocco, e cosi' via. L'ultimo blocco dati

di un file e' indicato da un primo byte che avra' un valore di $00 mentre il secondo byte conterra' il numero di Bytes utilizzati in quest'ultimo settore. Questa concatenazione potra' essere spiegata con l'aiuto del DOS MONITOR:

```
>:B0  A0 A0 A0 A0 A0 00 00 00       ...
>:B8  00 00 00 00 00 00 0B 00  ........
>:C0  00 00 81 13 09 54 31 32  .....T12
>:C8  2F 53 30 31 A0 A0 A0 A0  /S01
>:D0  A0 A0 A0 A0 A0 00 00 00       ...
>:D8  00 00 00 00 00 00 06 00  ........
>:E0  00 00 82 10 00 44 49 53  .....DIS
>:E8  4B 20 41 44 44 52 20 43  K ADDR C
>:F0  48 41 4E 47 45 00 00 00  HANGE...
>:F8  00 00 00 00 00 00 04 00
```

Quanto appena visto proviene dalla directory (traccia 18 settore 1) del dischetto TEST/DEMO che trovate insieme al floppy. Potete seguire l'organizzazione del file DISK ADDR CHANGE. L'ingresso di questo file incomincia al byte $E2 e termina con il byte $FF. Questo e' un file PRG cioe' un file programma che puo' essere riconosciuto con il tipo file $82 nel byte $E2. Questo file comprende quattro blocchi su dischetto. Cio' si puo' notare leggendo i byte da $FE e $FF. Vediamo ora una sezione di questo blocco, cioe' del blocco presente la traccia 16 settore 0:

```
>:00  10 0A 01 04 0F 04 64 00  ......$.
>:08  97 35 39 34 36 38 2C 31  .59468,1
>:10  32 00 39 04 6E 0D 99 22  2.9....".
>:18  93 13 11 11 11 11 44 52  ......DR
>:20  49 56 45 20 41 44 44 52  IVE ADDR
>:28  45 53 53 20 43 48 41 4E  ESS CHAN
>:30  47 45 20 50 52 4F 47 52  GE PROGR
>:38  41 4D 22 00 59 04 6F 00  AM".Y./.
>:40  99 22 11 54 55 52 4E 20  .".TURN
>:48  4F 46 46 20 41 4C 4C 20  OFF ALL
```

Questo blocco contiene la prima parte del programma. Si puo' notare che e' immagazzinato sul dischetto esattamente come e' immagazzinato nella memoria del computer.
I comandi basic infatti sono convertiti in un byte chiamato tokens (per il processo detto di TOKENIZZAZIONE, vedi il volume GUIDA AL CBM 64 Ed. EVM) allo stesso modo che avviene nella memoria interna del computer. In questo modo si occupera' meno posto e solo il testo sara' convertito in codici esadecimali. I primi due bytes di questo blocco dati indicano il secondo blocco dati $10 e $0A cioe' la traccia 16 settore 10. Vediamo questa sezione:

```
>:00   10 14 34 30 00 1D 05 A0   ..40...
>:08   00 8D 20 33 30 30 3A 20   .. 300:
>:10   8F 20 46 49 4E 44 20 44   . FIND D
>:18   52 49 56 45 20 54 59 50   DRIVE TYP
>:20   45 00 39 05 AA 00 8D 20   E.9.  ..
>:28   36 30 30 3A 20 8F 20 43   600: . C
>:30   48 41 4E 47 45 20 41 44   HANGE AD
>:38   44 52 45 53 53 00 68 05   DRESS.(.
>:40   B4 00 99 22 11 54 48 45    ..".THE
>:48   20 53 45 4C 45 43 54 45    SELECTE
```

Il programma continua in questo blocco. I primi due bytes contengono gli indirizzi del terzo blocco dati del file ($10, $14, cioe' traccia 16 settore 20):

```
>:00   10 08 31 30 30 30 00 23   ..1000.#
>:08   06 54 01 8B 20 43 B2 32   .T.. C 2
>:10   35 34 20 A7 20 4D 54 B2   54   MT
>:18   31 31 39 3A 20 8F 3A 20   119: .:
>:20   32 30 33 31 20 56 32 2E   2031 V2.
>:28   36 00 45 06 5E 01 8B 20   6.E. ..
>:30   43 B2 32 32 36 20 A7 20   C 226
>:38   4D 54 B2 35 30 3A 20 8F   MT 50: .
>:40   3A 20 32 30 34 30 20 56   : 2040 V
>:48   31 2E 32 00 67 06 68 01   1.2. .(.
```

Questo e' il blocco prima dell'ultimo blocco di programma. Avrete senza dubbio notato che i blocchi di dati sono nelle stessa traccia ma non sono contigui. Il primo blocco dati e' il blocco 0, il successivo e' il blocco 10, 10 blocchi dal primo blocco.
Nove blocchi sono sempre saltati tra blocchi di dati di un file. Il terzo blocco di dati e' il blocco n. 20. Il sistema operativo rincomincia con il primo blocco se il blocco calcolato supera il numero piu' alto di blocco presente nella traccia. Poiche' la traccia 16 contiene 21 blocchi l'ultimo blocco di dati e' il blocco n. 8. I primi due bytes dell'indirizzo di questo terzo blocco sono:

```
>:00   00 F8 5A 42 B2 31 20 A7   . ZB 1
>:08   20 34 34 30 00 14 07 A3    440...
>:10   01 8B 20 53 54 20 A7 20   .. ST
>:18   31 30 30 30 00 45 07 B8   1000.E.
>:20   01 98 31 35 2C 22 4D 2D   ..15,"M-
>:28   52 22 C7 28 31 37 32 29   R" (172)
>:30   C7 28 31 36 29 3A A1 23    (16): #
>:38   31 35 2C 5A 43 24 3A 5A   15,ZC$:Z
>:40   43 B2 C6 28 5A 43 24 AA   C F(ZC$
>:48   C7 28 30 29 29 00 66 07   G(0)).&.
```

Qui si vede che la fine del programma e' segnata dal valore $00 nel byte $00. Il byte $01 da il numero di bytes in questo ultimo blocco ($F8 corrisponde a 248 bytes). Vediamo ora di trovare la grandezza del programma:

```
3 blocchi con 254 bytes cad=762 bytes
l'ultimo blocco            =248 bytes
                           ----------
totale                     1100 bytes
```

che sara' la grandezza del programma

IL NOME DEL FILE

Come abbiamo accennato il nome del file e' contenuto nei bytes da 3 a 18 della lista del file entry. Consiste di un massimo di 16 caratteri e come abbiamo detto se il nome del file e' inferiore a 16 caratteri il resto dello spazio che dovrebbe essere occupato e' riempito con spazi shiftati del valore $AO.

TRACCIA E SETTORE DI UN NUOVO FILE PER LA SOVRASCRITTURA

Se un file viene sovrascritto utilizzando il comando ò:, per prima cosa il nuovo file viene completamente salvato.
Nessun nome di file viene scritto nella directory per questo file perche' il file stesso esiste gia' sotto lo stesso nome. Invece l'indirizzo del primo blocco del nuovo file e' immesso nei bytes 26 e 27 dell'entry file. Viene quindi cancellato il vecchio file e aggiornata la BAM. Quindi gli indirizzi relativi al primo blocco dati del nuovo file vengono immessi all'entry file nei bytes 1 e 2

NUMERO DI BLOCCHI IN UN FILE

La lunghezza di un file e' data nei bytes 28 e 29 dell'entry file. Un file consiste di almeno un blocco ed al massimo puo' essere di 664 blocchi, cioe' tutta la grandezza di un disco. Il primo byte e' il byte basso mentre il secondo e' il byte alto. Se per esempio scoprite attraverso il DISK MONITOR che un file e' di

lunghezza $1F,$00 il file sara' lungo 31 blocchi.

L'ORGANIZZAZIONE DI UN FILE RELATIVE

La differenza fondamentale fra un file sequenziale e un file relative consiste nel fatto che nei relative ogni record di dati puo' essere ritrovato tramite un numero di record. Il sistema operativo su disco o DOS del 1541 esegue la maggior parte delle operazioni richieste per la manipolazione di un file relative anche se i relative non sono previsti nel basic del COMMODORE 64 . Vediamo ora come viene organizzato un file relative tenendo presente che gli esami che noi esseguiremo saranno fatti in gran parte con il DISKMON. Per prima cosa apriamo un file relative con la lunghezza record di 100:

```
OPEN 2,8,2, "REL-FILE,L,"+CHR$(100)
```

Scriviamo ora dei dati nel numero di record 70:

```
OPEN 1,8,15
PRINT#1,"P"+CHR$(2)+CHR$(70)+CHR$(0)+(CHR$(1)
PRINT#2,"DATI PER IL RECORD N.70"
CLOSE 2 : CLOSE 1
```

Esaminando la directory con il DISKMON avremo quanto segue:

```
>:00 .. .. 84 11 00 52 45 4C   ...REL
>:08 2D 46 49 4C 45 A0 A0 A0 -FILE
>:10 A0 A0 A0 A0 A0 11 0A 64       ..$
>:18 00 00 00 00 00 00 1D 00 ........
```

Il primo byte $84 specifica che siamo in presenza di un file relative. I successivi due bytes indicano la prima traccia e settore dei data scritti ($11,$00 che equivalgono alla traccia 17 settore 0) esattamente come su un file sequenziale. Come abbiamo visto in precedenza segue il nome del file (16 caratteri, seguiti per totale riempimento dagli spazi shiftati, $A0. Di seguito ci sono due campi non utilizzati con i files sequenziali. Il primo campo e' un puntatore di due bytes, alla traccia e al settore del primo blocco SIDE-SECTOR. Un side-sector contiene i puntatori a ogni record di data. Lo descriveremo in dettaglio successivamente ($11,$0A; traccia 17, settore 10). Il secondo campo e' di un byte che contiene la lunghezza del record, valore tra 1 e 254, nel nostro caso $64 (100). Il vantaggio di essere in grado di poter accedere ad ogni record individualmente richiede una definita lunghezza per ogni record che deve essere dichiarata quando si crea un file relative. I restanti campi nella directory hanno i normali significati. Gli ultimi due bytes contengono il numero di blocchi nel file (byte alto e basso cioe' $1D e $00).
Vediamo ora quali sono le funzioni del side-sector e che cosa contiene.
I blocchi di side-sector contengono i puntatori alla traccia e settore di ogni singolo records di dati. Per esempio se desideriamo leggere il settantesimo record di un file relative il DOS consulta il blocco di side sector per sapere quale traccia e settore contengano il record e quindi legge il record direttamente. Il risultato di questo e' che possiamo leggere il settantesimo record di un file senza aver letto prima l'intero file. Vediamo ora l'esatta costruzione di un blocco di side-sector.

UTILIZZO DEI SIDE SECTOR

Quanto segue e' stato come al solito ottenuto tramite il monitor del disco.

```
>:00  00 47 00 64 11 0A 00 00  .G.$....
>:08  00 00 00 00 00 00 00 00  ........
>:10  11 00 11 0B 11 01 11 0C  ........
>:18  11 02 11 0D 11 03 11 0E  ........
>:20  11 04 11 0F 11 05 11 10  ........
>:28  11 06 11 11 11 07 11 12  ........
>:30  11 08 11 13 11 09 11 14  ........
>:38  10 08 10 12 10 06 10 10  ........
>:40  10 04 10 0E 10 02 10 0C  ........
>:48  00 00 00 00 00 00 00 00  ........
>:50  00 00 00 00 00 00 00 00  ........
 etc.
```

Come di norma i primi due bytes puntano alla traccia e settore del successivo blocco di side-sector.
Nel nostro caso non esistono ulteriori blocchi di side-sector ($00) e solo $47 = 71 bytes di questo settore sono utilizzati. Il byte 2 contiene il numero del blocco di side-sector, 00. Un file relative puo' contenere fino a un massimo di sei di questi blocchi, la numerazione va da 0 a 5.
La lunghezza del record, $64 (100) si rileva nel byte 3.
I prossimi 12 bytes (dal byte 4 fino al byte 15) contengono i puntatori alla traccia e settore (2 bytes ognuno) riferentesi ai 6 blocchi di side-sector (00,00 sta a significare che il blocco non e' ancora utilizzato).
Partendo dal byte 16 ($10) ci sono i puntatori ai data e i puntatori traccia e settore ai primi 120 blocchi di

dati (nel nostro caso solo 28 puntatori).
Utilizzando il numero di record e la lunghezza del record il DOS puo' calcolare in quale blocco sono presenti i data e a quale posizione entro il blocco inizia il record. Per comprendere meglio facciamo questo esempio:
Per leggere il settantesimo record dal file con una lunghezza di record di 100 caratteri possiamo eseguire il seguente calcolo:

(70-1) * 100 / 254

Avremo un quoziente di 27 con resto di 42.
Il DOS ora sa che il record puo' essere trovato nel ventisettesimo blocco dati alla 42+2 o 44esima posizione.
Vediamo ora una spiegazione del calcolo.
Ogni blocco contiene 256 bytes, i primi due del quale sono utilizzati come puntatori per il blocco successivo. Sono quindi lasciati liberi per l'immagazzinamento di dati 254 bytes. Si puo' calcolare il numero del byte dall'inizio del file (che e' il record 1) partendo dal numero del record e dalla sua lunghezza. Se noi dividiamo questo valore per il numero di bytes per blocco avremo il numero del blocco che contiene il record. Il resto della divisione ci da la posizione all'interno del blocco (aggiungere 2 perche' i primi 2 bytes servono come abbiamo visto per puntare al blocco successivo).
Se il record supera la fine del blocco deve essere letto anche il blocco successivo. Nel nostro esempio il 27esimo blocco di dati si trova nella traccia $10 (16) e nel settore $0C (12). Se noi leggiamo questo blocco con il monitor avremo i seguenti dati:

```
>:00  00 F3 00 00 00 00 00 00   ........
>:08  00 00 00 00 00 00 00 00   ........
>:10  00 00 00 00 00 00 00 00   ........
>:18  00 00 00 00 00 00 00 00   ........
>:20  00 00 00 00 00 00 00 00   ........
>:28  00 00 00 00 44 41 54 41   ....DATA
>:30  20 46 4E 52 20 52 45 43    FOR REC
>:38  46 52 44 20 37 30 0D 00   ORD 70..
>:40  00 00 00 00 00 00 00 00   ........
>:48  00 00 00 00 00 00 00 00   ........
>:50  00 00 00 00 00 00 00 00   ........
>:58  00 00 00 00 00 00 00 00   ........
>:60  00 00 00 00 00 00 00 00   ........
>:68  00 00 00 00 00 00 00 00   ........
>:70  00 00 00 00 00 00 00 00   ........
>:78  00 00 00 00 00 00 00 00   ........
>:80  00 00 00 00 00 00 00 00   ........
>:88  00 00 00 00 00 00 00 00   ........
>:90  FF 00 00 00 00 00 00 00   ........
>:98  00 00 00 00 00 00 00 00   ........
>:A0  00 00 00 00 00 00 00 00   ........
>:A8  00 00 00 00 00 00 00 00   ........
>:B0  00 00 00 00 00 00 00 00   ........
>:B8  00 00 00 00 00 00 00 00   ........
>:C0  00 00 00 00 00 00 00 00   ........
>:C8  00 00 00 00 00 00 00 00   ........
>:D0  00 00 00 00 00 00 00 00   ........
>:D8  00 00 00 00 00 00 00 00   ........
>:E0  00 00 00 00 00 00 00 00   ........
>:E8  00 00 00 00 00 00 00 00   ........
>:F0  00 00 00 00 FF 00 00 00   ........
>:F8  00 00 00 00 00 00 00 00   ........
```

Se dal calcolo otteniamo un numero di blocco maggiore di 120 il puntatore non puo' essere trovato nel primo blocco di side-sector, ma nei successivi blocchi di side-sector.
In questo caso sì divide il numero blocco per 120 e il quoziente sara' il numero del blocco di side-sector. Il resto della divisione ci dara' l'indirizzo del puntatore entro questo blocco.
Dovendo per esempio trovare il record n. 425 eseguiremo la divisione per 120 come ho detto e avremo un quoziente di 3 e un resto di 65. Percio' dovrete leggere il blocco side-sector 3 e inviare il puntatore al 65esimo blocco data.
Vediamo cosa succede quando si crea o si espande un file relative.
Per prima cosa viene creato un DIRECTORY ENTRY per il file reletive che contiene la lunghezza del record.
Sono poi riservati due canali per i file relative, uno per i data, l'altro per i side-sectors.
Se un puntatore di record e' fissato a un dato record, il DOS per prima cosa controlla per vedere se il record esiste gia'. Se cosi' e' viene letto il corrispondente blocco e il BUFFER POINTER fissato in maniera tale che si possa accedere al contenuto del blocco. Se non e' cosi' il record viene creato.
Tutti i record precedenti questo numero di record che non e' ancora presente sono anch'essi creati. Il primo byte di un nuovo record scritto contiene $ff (255), e il resto del record viene riempito con $00.
Se il corrispondente record e' all'inizio di un blocco il resto del blocco viene riempito con record vuoti. Ogni volta che si tenta di accedere a un record inesistente viene visualizzato il messaggio di errore:

50,RECORD NOT PRESENT.

Quando si scrive un nuovo record questo non e' da considerarsi un errore, ma indica che un nuovo record e' stato creato.
Potete utilizzare questo metodo per creare nuovi files se conoscete il numero massimo di records. Sara' sufficiente fissare il puntatore a questo record e scrivere $FF (CHR$(255)) in questo record. Allocando un file come questo il messaggio di errore non apparira' piu'. Dovete anche essere a conoscienza se c'e' abbastanza spazio sul dischetto. Se non c'e' verra' visualizzato un messaggio di errore:

52, FILE TOO LARGE

Come massimo di side-sectors un file relative puo' contenere 6 * 120 * 254 = 182880 bytes.
Nel caso del 1541 questo e' piu' della capacita' dell'intero dischetto. Per quanto riguarda invece gli altri tipi di DOS e' bene controllare il libro "LE PERIFERICHE COMMODORE".
Poiche' un file relative richiede tre canali di dati e il 1541 ha solo cinque canali disponibili, solamente un file relative per volta puo' essere aperto. Gli altri due canali disponibili potranno essere utilizzati per tener aperto un file sequenziale allo stesso tempo.
Naturalmente con i dischi COMMODORE piu' grandi tipo 8050 e 8250 si possono tenere aperti un numero maggiore di files relatives contemporaneamente.

CAPITOLO SESTO

ACCESSO DIRETTO AD UN QUALSIASI BLOCCO DEL DISCHETTO.

Quando si manipolano files e programmi su un dischetto, non dobbiamo eseguire nessuna operazione riguardo all' organizzazione dei dati sul dischetto stesso, in quanto esiste il SISTEMA OPERATIVO SU DISCO (DOS), che si prende cura di ogni e qualsiasi dettaglio relativo a questa organizzazione.
Tuttavia il DISK OPERATING SYSTEM offre la possibilita' di accedere ad ogni blocco, o piu' precisamente ad ogni informazione sul dischetto.
Cio' rende molto flessibile la manipolazione dei singoli files per creare strutture di dati completamente nuove rispetto a quele tradizionalmente utilizzate.

Per accedere direttamente ad un blocco deve essere aperto un canale sul BUFFER dati del drive, e sara' quindi su questo canale che i dati verranno trasmessi.
Il BUFFER di dati serve come immagazzinamento intermedio o temporaneo per dati che devono essere letti o scritti sul dischetto.
Ovviamente e' necessario informare il SISTEMA OPERATIVO DISCO che si sta lavorando con comandi ad accesso diretto e per questo utilizzeremo uno speciale nome di file nel comando di apertura (OPEN):

```
OPEN1,8,2,"#"
```

Utilizzando questo comando, il file logico numero 1

sulla periferica 8 (il drive) viene associato con un file ad accesso diretto.
Il canale numero 2 serve per la trasmissione dati al e da disco.
Il numero di canale, cioe' l' indirizzo secondario nel comando OPEN, deve essere compreso nell' intervallo fra i valori 2 e 14.
I canali 0 e 1 sono infatti riservati per i comandi LOAD e SAVE, mentre il canale 15 e' il canale di comando.
La scelta di un Indirizzo secondario e' arbitraria.
Non si deve mai utilizzare lo stesso indirizzo secondario simultaneamente, cioe' con 2 comandi, perche' il SISTEMA OPERATIVO DISCO non appena dovesse incontrare un secôndo comando OPEN con lo stesso indirizzo secondario chiudera' il file al quale ci stiamo riferendo, utilizzando questo numero di canale.

Cio' accade anche quando si sta lavorando con files RELATIVES o ad accesso sequenziale.

Questa forma del comando OPEN consente che il SISTEMA OPERATIVO DISCO ricerchi un BUFFER libero di dati e gli assegni quel determinato canale.
Utilizzando un comando GET# immediatamente dopo l' OPEN si puo' trovare il numero di BUFFER che il sistema operativo ha assegnato a questa operazione:

```
100 OPEN 1,8,2,"#"
110 GET#1,A$
120 PRINT ASC(A$+CHR$(0))
```

```
RUN
```

verra' visualizzato il numero 3

In questo caso cioe' e' stato assegnato il BUFFER numero 3.

I BUFFER sull' unita' a dischi 1541 sono 5 e vengono convenzionalmente numerati da 0 a 4.
Ogni Buffer puo' manipolare fino a 256 bytes di dati.
Di seguito riportiamo gli indirizzi di memoria dei 5 BUFFER ricordando di non confonderli con gli indirizzi relativi all' unita' centrale.

BUFFER	INDIRIZZI	ESA/DEC
0	$300-3FF	768-1023
1	$400-4FF	1024-1279
2	$500-5FF	1280-1535
3	$600-6FF	1536-1791
4	$700-7FF	1792-2047

Il BUFFER numero 4 normalmente non e' disponibile perche' vi e' immagazzinata la BAM o BLOCK AVAILIBILITY MAP del disco.
Se si lavora con files sequenziali e relatives allo stesso tempo anche il BUFFER numero 3 non e' disponibile perche' utilizzato per la DIRECTORY.
Se si ritiene necessario associare ad un comando disco uno specifico BUFFER per l' accesso diretto, questa operazione dovra' essere fatta utilizzando un comando OPEN come nell' esempio seguente:

```
OPEN1,8,2,"#3"
```

Questo comando OPEN assocera' il BUFFER numero 3 ($600-$6FF) con il canale di comunicazione numero 2, partendo naturalmente dall' ipotesi che questo sia libero.

NOTA

A meno che non abbiate una ragione molto importante per utilizzare uno specifico Buffer, dovreste lasciare questa. scelta, cioe' quella relativa al buffer da utilizzare,al SISTEMA OPERATIVO DISCO stesso poiche' questa scelta sara' comunque fatta per un BUFFER realmente disponibile.

Dopo aver aperto un canale dovete ricordarvi di controllare il canale di errore, ad esempio con questi comandi:

```
130 OPEN 15,8,15
140 GET#15,A$ : PRINTA$;: IF ST<>64 THEN 140
```

Se il BUFFER e' in quel momento utilizzato riceverete il messaggio di errore:

```
70, NO CHANNEL,00,00
```

Se non ci sono altri files aperti potrete aprire fino a

quattro canali per l' accesso diretto.
Il seguente esempio illustra quanto detto:

```
10 OPEN 1,8,15,"I0": I=2
20 OPEN 2,8,2, "#": GOSUB 100
30 OPEN 3,8,3, "#": GOSUB 100
40 OPEN 4,8,4, "#": GOSUB 100
50 OPEN 5,8,5, "#": GOSUB 100
60 OPEN 6,8,6, "#": GOSUB 100
70 END
100GET#I,A$:PRINT ASC(A$+CHR$(0))
110I=I+1
120GET#1,A$:PRINTA$;:IF ST<>64 THEN 120
130RETURN
```

Quando il precedente programma gira, cioe' dopo aver dato il RUN, avremo la seguente visualizzazione:

```
      3

00,OK,00,00
      2
00,OK,00,00
      1
00,OK,00,00
      0
00,OK,00,00
    199
70,NO CHANNEL,00,00
```

Come si puo' osservare il tentativo di aprire un quinto canale per l' accesso diretto, non dara' nessun risultato.

La trasmissione di dati a e da un BUFFER normalmente viene eseguita utilizzando dei comandi GET#, INPUT# e PRINT#.
Se un BUFFER contiene solamente testi,cioe' solamente dati alfanumerici, che non abbia una lunghezza maggiore di 88 caratteri e sia separato, abbia cioe' come carattere di separazione,un ritorno carrello (CHR$(13)), puo' essere letto utilizzandoun comando INPUT#.
Tuttavia se il BUFFER contiene caratteri di controllo o nel testo stesso sono stati utilizzati dei caratteri virgola o punto e virgola (, o ;), allora il comando INPUT# non va bene.
In questo caso dovremo utilizzare un comando GET# che eseguira' l' operazione su un solo carattere per volta.

NOTA

Ricordiamo pero' che il GET# non consente di leggere valori nulli cioe' CHR$(0). In questo caso il GET# riceve una stringa vuota e dovrete quindi controllare l' eventualita' di questa condizione con la seguente metodologia:

```
100 GET#2, A$ : IF A$="" THEN A$ = CHR$ (0)
```

Una semplice alternativa al comando GET# e' quella di utilizzare un comando INPUT* come vedremo in seguito.
Con questo sistema potrete dichiarare quanti caratteri devono essere letti nellastringa.
Questo comando puo' anche manipolare valori nulli come CHR$(0), e si puo' leggere anche l' intero BUFFER di 256 caratteri con un solo comando.

Nel seguente capitolo, tutti i comandi ad accesso diretto verranno dettagliatamente descritti.

Quando si utilizzano comandi ad accesso diretto, si deve esplicitamente consentire che i blocchi sul dischetto possano essere letti o scritti.
I comandi ad accesso diretto sono inviati sul canale di comando numero 15.

I dati che vengono scritti o letti su un BUFFER sono trasmessi su un canale separato che e' associato con quel Buffer.
Sia il canale numero 15 che il canale associato con il BUFFER devono essere aperti prima che inizi la trasmissione di dati in un verso o nell' altro.

Riepilogando:

1)Un comando PRINT# al canale di comando numero 15 invia un ordine di accesso diretto al SISTEMA OPERATIVO DISCO.

2)Un comando PRINT# ai canali da 2 a 14 invia dati ad un BUFFER.

3)Un comando INPUT# o GET# al canale di comando 15 riportano un qualsiasi messaggio di errore controllato dal DOS.

4)Un comando GET# o INPUT# ai canali da 2 a 14 leggono i dati dal BUFFER.

COMANDI DI ACCESSO DIRETTO.

BLOCK-READ B-R

Un comando di BLOCK-READ comunica al 1541 di leggere un blocco dal dischetto entro un dato BUFFER, di un file ad accesso diretto, preventivamente aperto.
Quest' ordine viene inviato sul canale di comando (indirizzo secondario 15) all' unita' a dischi.
Il comando puo' essere abbreviato con B-R.
Poiche' questo comando per sua natura non legge il primo byte del blocco e' necessario sostituirlo con U1 per leggere per intero il blocco.
Il comando ha la seguente sintassi:

```
U1;numero del canale;drive;traccia;settore
```

E' necessario fornire il numero di canale che si sta utilizzando quando viene eseguito un OPEN per accesso diretto sul file:

```
10 OPEN 1,8,15,"I0"
20 OPEN 2,8,2,"#"
30 PRINT#1,"U1;2;0;18;0"
```

Con questo esempio si legge il contenuto della traccia 18 del settore 0 entro il BUFFER associato con il canale 2.

Leggiamo ora i dati da questo BUFFER con un comando GET#2:

```
40 GET#2,A$,B$
50 PRINT ASC(A$), ASC(B$)

18    1
```

Abbiamo quindi letto e visualizzato i primi 2 Bytes del BUFFER.
Il settore 0 della traccia 18 contiene un puntatore al primo blocco della DIRECTORY ed alla BAM del dischetto.
Nel programma dimostrativo DISPLAY T & S fornito sul dischetto TEST demo, questo comando viene utilizzato per leggere la BAM dal disco e per visualizzare graficamente ogni record presente sul dischetto stesso.
Si possono leggere tutti e 256 bytes del blocco dal BUFFER con il comando GET#.
Nel nostro esempio leggeremo il nome del dischetto e l' identificatore dalla posizione 144.
I blocchi che compongono un file sono strettamente collegati l' uno all' altro.
I primi due bytes di ogni blocco contengono un puntatore al settore ed alla traccia del blocco successivo.
Utilizzando questa informazine si puo' agevolmente osservare l' utilizzo dello spazio su un dischetto per un file.
Un puntatore di traccia che contenga il valore 0 (zero) indica che siamo in presenza dell' ultimo blocco di un file. Mentre il puntatore che normalmente contiene il numero di settore, in questo caso conterra' il numero di Bytes dell' ultimo blocco che e' parte di questo file.
Il primo settore di un file puo' essere letto con il programma che vedremo successivamente e che si chiama

visualizzazione di tutti i parametri di un file.

Il seguente piccolo programma visualizza tutte le restanti tracce e settori che sono parte del file.

```
100 OPEN 1,8,15,"IO"
110 OPEN 2,8,2,"#"
120 INPUT"TRACCIA  E SETTORE";T,S
130 PRINT #1,"U1";2;0;T,S
140 GET#2, T$,S$
150 T=ASC(T$+CHR$(0)):S=ASC(S$+CHR$(0))
160 IF T=0 THEN CLOSE 2:CLOSE1:END
170 PRINT "TRACCIA";T,"SETTORE";S
180 GOTO130
```

NOTA

Immettere i valori 18 e 0 rispettivamente come traccia e settore per visualizzare la BAM e la DIRECTORY di quel dato dischetto.

BUFFER-POINTER   B-P

Il nome del dischetto inizia all' indirizzo 144 della traccia 18 del settore 0.
Utilizzando il precedente esempio possiamo leggere i primi 143 del BUFFER ed essere quindi posizionati all' inizio del nome del dischetto.
Ma il SISTEMA OPERATIVO DISCO ha una via piu' semplice per eseguire questa operazione.
Per accedere ad un qualsiasi necessario byte di un

buffer si puo' usare il comando di BUFFER-POINTER.
Usando questo comando il DOS ha la possibilita' di posizionarsi in un punto preciso all' interno del BUFFER.
Il comando BUFFER-POINTER puo' essere abbreviato con B-P.
La sua sintassi e' la seguente:

```
B-P;numero del canale; posizione
```

Possiamo ora leggere il nome del dischetto in maniera diretta:

```
100   OPEN   1,8,15,"IO"                             110   OPEN
2,8,2,"#"
120 PRINT#1,"U1";2;0;18;0
130 PRINT#1,"B-P";2;144
140 FORI=1 TO 16
141 REM LUNGHEZZA MASSIMA
150 GET#2,A$:IFA$=CHR$(160) THEN 170
160 PRINTA$;:NEXT
170 CLOSE 2:CLOSE 1
```

Con questo esempio, per prima cosa leggiamo il blocco, fissiamo quindi il puntatore del BUFFER alla posizione 144 e quindi leggiamo e visualizziamo il nome del dischetto che potra' avere una lunghezza massima di 16 caratteri.
Uno spazio SHIFTATO (CHR$(160)), indica la fine del nome del dischetto.
I Bytes nel buffer sono numerati da 0 a 255 e il primo Byte contiene un numero 0.

Il puntatore del BUFFER (BUFFER-POINTER) e' automaticamente messo a 0 (zero) durante la lettura di un blocco con il comando U1.
Si puo'per esempio leggere il Byte numero 2 dopo aver letto il nome del dischetto.
Per fare questa operazione sara' necessario fissare il puntatore del buffer a questo valore.
Esempio:

```
PRINT#1,"B-P";2;2
```

BLOCK-WRITE B-W

Il comando BLOCK-WRITE consente di scrivere il contenuto di un BUFFER su un dato blocco del dischetto.
Con cio' si puo' scrivere il blocco inviato al BUFFER entro l' unita' a dischi.
E' possibile leggere un blocco entro un BUFFER con un comando BLOCK-READ, cambiare il contenuto di alcuni Bytes, e quindi riscrivere il blocco.
Questo comando puo' essere abbreviato con le lettere B-W.
Poiche' questo comando scrive i contenuti del puntatore del BUFFER si puo' utilizzare il comando U2 che mettera' sempre a 1 il puntatore del BUFFER.
La sintassi del comando e' analoga a quella del BLOCK-READ:

U2 numero del canale;drive;traccia;settore

```
100 OPEN 1,8,15,"IO"
110 OPEN 2,8,2,"#"
120 PRINT#2," TEST DATA"
130 PRINT#1,"U2";2;0;1;0
140 CLOSE 2 :CLOSE 1
```

Nell' esempio riportato la frase "TEST DATA" sara' scritta sul Buffer associato al canale numero 2 quindi scritta sulla traccia 1 settore 0 del dischetto.

Il comando U2 fa si che non vengano eseguiti cambiamenti sui contenuti del BUFFER.

Vediamo ora un' esempio di utilizzo del comando BLOCK-WRITE per cambiare il nome del dischetto che avevamo letto con il precedente programma.
Per far questo sara' necessario riempire con un nuovo nome di 16 caratteri e con al termine CHR$(160) il Buffer cosi' che si possa scriverlo su disco.
Useremo ancora il comando BUFFER-POINTER per fissare il puntatore del buffer direttamemte alla richiesta posizione entro il BUFFER.

```
100 OPEN 1,8,15,"IO"
110 OPEN 2,8,2,"#"
120 PRINT#1,"U1";2;0;18;0
130 PRINT#1,"B-P";2;144
140 A$="NUOVO NOME FILE"
150 IF LEN(A$)¨16THEN A$=A$+CHR$(160):GOTO150
160 PRINT#2,A$;
170 PRINT#1,"U2";2;0;18;0
180 PRINT#1,"IO":CLOSE 2:CLOSE 1
```

Per prima cosa leggiamo la traccia 18 settore 0 entro il BUFFER. Fissiamo quindi i puntatori del BUFFER alla posizione del nome del dischetto e scriviamo un nuovo nome di 16 caratteri nel BUFFER.
Notare che il nome del dischetto e' stato cambiato solo sul BUFFER.
Tuttavia nella linea 170 del programma i contenuti del BUFFER sono scritti nello stesso blocco che cambia in misura permanente il nome del file sul dischetto.
Successivamente il dischetto e' inizializzato cosi' che la BAM ed il nome nella memoria del SISTEMA OPERATIVO DISCO sia aggiornato.

Esaminare ora la directory del dischetto con:

```
LOAD"$",8
LIST
```

per vedere se effettivamente e' stata apportata la variazione richiesta.

BLOCK-ALLOCATE B-A

Questo comando ha il compito di indicare se nella BAM esiste un particolare blocco che sta per essere usato.
Il comando BLOCK-ALLOCATE puo' essere abbreviato con B-A.
Per i programmi, per i files sequenziali o relatives non appena i blocchi del dischetto sono utilizzati,la BAM viene aggiornata in modo tale da far sapere al SISTEMA OPERATIVO DISCO che quei blocchi non sono piu' disponibili.

Tuttavia i blocchi scritti utilizzando comandi ad accesso diretto non sono automaticamente allocati.
Quando blocchi utilizzati in questa maniera non sono correttamente e definitivamente allocati esiste la possibilita' che essi vengano sovrascritti qualora si usino altri files.
Il comando BLOCK-ALLOCATE deve essere utilizzato per prevenire questo tipo di sovrascrittura.
Questo comando ha la seguente sintassi:

```
B-A;drive;traccia;settore
```

Con cio' il corrispondente blocco nella BAM viene segnato come blocco allocato ed e' protetto contro la sovrascrittura di altri files.
Se si tenta di scrivere nuovamente su di un blocco correttamente allocato, il canale di errore restituira' un messaggio di tipo:

```
65, NO BLOCK
```

```
100 OPEN 1,8,15,"IO"
110 INPUT"TRACCIA, SETTORE";T,S
120 PRINT#1,"B-A";0;T;S
130 INPUT#1,A$,B$,C$,D$
140 PRINTA$","B$","C$","D$
```

Utilizzando questo programma si puo' immettere un numero di traccia e settore di un blocco che si desidera allocare.
Se il blocco e' tuttora libero questo sara' allocato ed

il messaggio:

    00,OK,00,00

viene visualizzato.
Mentre se il blocco e' gia' allocato verra' visualizzato il messaggio:

    65,NO BLOCK,TRACCIA,SETTORE

In questo caso i parametri TRACCIA e SETTORE contengono il successivo numero di blocco libero sul dischetto.
Cio' sta a significare che il richiesto blocco e' allocato, ma il blocco il cui indirizzo e' dato appunto dai parametri TRACCIA,SETTORE e' ancora disponibile.
Se il messaggio di errore visto in precedenza riporta ZERO come valori di traccia e settore, cio' stara' a significare che nessun blocco con un piu' alto numero di traccia e settore e' disponibile.

Il seguente programma provvede all' allocazione automatica dei successivi settori liberi.

```
100 OPEN 1,8,15,"IO"
110 INPUT"TRACCIA, SETTORE";T,S
120 PRINT#1,"B-A";0;T;S
130 INPUT#1,A$,B$,TT,SS
140 IFA$="00"THEN 190
150 IFA$<>"65".THENPRINTA$",“B$","TT","SS:GOTO300
160 IFTT=0THENPRINT"NESSUN ALTRO B.LIBERO":GOTO300
170 IF TT=18 THEN TT=19:SS=0
```

```
180 T=TT:S=SS:GOTO 120
190 PRINT"TRACCIA"TT"SETTORE"SS"ALLOCATI"
300 PRINT#15,"IO":END
```

Il controllo per la traccia 18 nella linea di programma 180 previene che un blocco nella DIRECTORY possa essere allocato.
L' impiego del comando B-A puo' riportare un' altro tipo di errore che e' interessante esaminare.
Se viene eseguito un tentativo di allocare un blocco che non puo' esistere, ad esempio traccia 20 settore 21, verra' visualizzato un messaggio di errore:

66, ILLEGAL TRACK OR SECTOR,20,21

Come abbiamo detto, il fatto di segnare un blocco come allocato nella BAM, previene il fenomeno che venga sovrascritto da altri files.
Il blocco sara' riconosciuto come allocato fino a che non sia inviato un comando di VALIDATE oppure, (solo pero' per il BASIC 4.0) un comando di COLLECT.
Il comando VALIDATE ricostruisce una nuova BAM eseguendo un' operazione di concatenamento dei blocchi nel singolo file e marcando ogni blocco in maniera tale che lanuova BAM venga ricostruita partendo da zero.

NOTA

Con il VALIDATE i files non correttamente chiusi e che all' esame della directory appaiono segnati con un asterisco (*) saranno cancellati.

BLOCK-FREE B-F

Questo comando, che puo' essere abbreviato con B-F, esegue la funzione opposta a quella appena vista nel comando precedente.
In sostanza segnala un blocco come NON ALLLOCATO nella BAM.
La sintassi e' molto simile a quella appena vista:

B-F drive traccia settore

esempio:

```
100 OPEN 1,8,15
110 PRINT#1,"B-F";0;20;9
```

In questo caso il blocco nella traccia 20 settore 9 viene reso disponibile nella BAM. Se il blocco e' gia' disponibile non viene riportato nessun errore.
I comandi BLOCK-WRITE e BLOCK-READ non controllano la BAM prima di sovrascrivere dei blocchi .
Con questi comandi potete scrivere su blocchi segnati come se in effetti fossero correttamente allocati nella BAM.
Se per esempio avete un disco che contiene solo files ad accesso diretto all' inizio non e' necessario allocare blocchi scritti perche' nessun altro file sara' scritto su dischetto. In questo caso si puo' usare la DIRECTORY dei blocchi nela traccia 18 ed avere 672 blocchi disponibili sul 1541.

BLOCK-EXECUTE B-E

Il comando BLOCK-EXECUTE consente che un blocco possa essere letto da un dischetto, inserito in un buffer e quindi i contenuti del buffer siano eseguiti come programma in linguaggio macchina.
Si possono scrivere routines per il SISTEMA OPERATIVO DISCO, memorizzarle con un comando B-W o U2 ad un settore e successivamente caricarle entro un BUFFER e mandarle in esecuzione con un comando BLOCK-EXECUTE con cui verranno eseguite come programmi in linguaggio macchina.
Naturalmente cio' presuppone la conoscenza del lavoro che il DOS eseguira' all' interno.
Se desiderate usare un comando B-E, normalmente dovrete dare il numero del BUFFER da utilizzare nel comando OPEN nel caso che il programma in linguaggio macchina non sia rilocabile e sia scritto per un dato BUFFER.
Il comando BLOCK-EXECUTE ha la seguente sintassi:

```
B-E;n.canale;drive;traccia;settore
```

esempio:

```
100 OPEN 1,8,15,"IO"
110 OPEN 2,8,2,"#3"
120 PRINT#1,"B-E";2;0;17;12
```

In questo esempio il BUFFER 3 (da $600 a $6FF) e' assegnato al canale 2.
I contenuti della traccia 17 settore 12 sono caricati all' interno di questo BUFFER e qui vieneeseguito il programma in linguaggio macchina.

Il comando BLOCK-EXECUTE e' una combinazione di comandi BLOCK-READ e MEMORY-EXECUTE.
Nella parte successiva del manuale sono forniti esempi di programmi in linguaggio macchina che possono essere eseguiti dal DOS tramite dei comandi cosidetti di memoria.

UTILIZZO DEGLI ACCESSI DIRETTI

Vediamo un attimo che cosa consentono di fare i comandi ad accesso diretto.
Eseguendo delle manipolazioni sui singoli settori si possono effettuare dei cambiamenti nella zona riservata alla BAM (traccia 18 settore 0) allo stessomodo incui sipuo' cambiare ilnome del dischettoo l'identificatore.
Si possono effettuare cambiamenti nella DIRECTORY che come ricordiamo inizia dalla traccia 18 settore 1.
Ricordiamo che ogni ogni ingresso di file nella DIRECTORY ha uno spazio di file non utilizzato. Si puo' utilizzare quindi questo spazio per immagazzinare informazioni addizionali.
Si possono cambiare i nomi dei files nella directory utilizzando comandi ad accesso diretto.
Si puo' ad esempio seguire il cambiamento dei blocchi in un file per vedere se il file stesso e' intatto.
Si puo' chiudere un file restato aperto mettendo a 0 il bit 7 dell7 indicatore di TIPI FILE nella directory.
Per esempio sipuo' cambiare l' indicatore del tipo del file da $02 a $82.
Normalmente questi files sono indicati nella DIRECTORY con un' asterisco.
Dopo il cambiamento l' asterisco sparira'.

Ogni file entry contiene anche una serratura che disabilita la possibilita' di cancellarlo.
Se viene settato il bit 6 relativo al tipo del file allora viene inviata l' informazione che quel file non e' disponibile per la cancellazione, cioe' non puo' essere cancellato.
Con questo memodo si possono proteggere programmi e dati da cancellazioni accidentali.

Altro esempio dell' utilizzo dell' accesso diretto puo' essere puo' essere con la cancellazione di programmi e files da dischetto. Se non e' stato scritto nient' altro su quella zona del dischetto, o meglio ancora non e' stato scritto niente del tutto sul dischetto, potra' essere possibile recuperare un file accidentalmente cancellato .
Cio' perche' la cancellazione di un file viene fatta semplicemente mettendo a 0 il tipo del file nella DIRECTORY e rendendo quindi liberi i blocchi allocati.
Sara' quindi necessario cercare solo nella lista della Directory il nome del file che si vuole recuperare e ripristinare il tipo di file:

$81 per i files sequenziali
$82 per i files programma
$83 per i files USR
$84 per i relatives

Dopo aver ripristinato iltipo di file dovrete usare il comando VALIDATE per riallocare nuovamente i blocchi.
Ad esempio:

```
OPEN 1,8,15:PRINT#1,"VO"
```

Altro uso di comandi ad accesso diretto puo' essere

quello di implementare nuove strutture di dati che il DOS normalmente non riconosce.
Ad esempio puo' essere creata una struttura di dati di tipo ISAM, che e' un' abbreviazione di INDEX SEQUENTIAL ACCESS METHOD, cioe' metodo di accesso sequenziale indicizzato.
Con un file di tipo ISAM si puo' accedere direttamente ad ogni record allo stesso modo di come si opera con unfile relative.
Tuttavia questo accesso non e' per numero di record ma per chiave o per indice.
Questo indice sara' un campo entroil record. Se per esempio un record e' composto da 5 campi:

COGNOME
NOME
INDIRIZZO
CITTA'
CODICE DI AVVIAMENTO POSTALE

Il cognome puo' essere definitocomechiave di accesso.
Per leggere quindi il record ROSSI il comando sara' semplicemente:

leggi il record ROSSI.

In questi tipi di files l' indice e' di norma salvato separatamente insieme alle informazioni relative al punto in cui i records possono essere trovati su disco.

COME ACCEDERE AL DOS. I COMANDI MEMORY

Nel precedente capitolo abbiamo esaminato il problema di caricare un programma nella memoria DOS e di eseguirlo.
Con i comandi cosi' detti di memoria (MEMORY) si puo' accedere invece al singolo BYTE del SISTEMA OPERATIVO DISCO ed eseguire programmi in RAM o in ROM.
Per esempio si puo' accedere al WORK SPACE o spazio di lavoro del SISTEMA OPERATIVO DISCO e leggere il numero di blocchi liberi su disco o prelevare il nome del disco dal BUFFER della BAM.
Avendo la possibilita' di scrivere sulla memoria RAM del SISTEMA OPERATIVO DISCO si possono cambiare delle costanti come il numero della periferica o il numero dei tentativi di lettura di un blocco.
Si possono utilizzare per questo anche le routines presenti nella memoria del SISTEMA OPERATIVO DISCO.
Per fare questo naturalmente e' necessaria una notevole conoscenza sia del SISTEMA OPERATIVO DISCO sia del linguaggio ASSEMBLER o del linguaggio macchina del 6502/6510 e del loro sistema di operare.

MEMORY-READ M-R

Utilizzando questo comando si puo' accedere ad ogni Byte del SISTEMA OPERATIVO DISCO. Questo comando puo' essere abbreviato con M-R.
Il comando MEMORY-READ e' trasmesso sul canale di comando. Il Byte letto e' quindi riportato sul canale di comando dove puo' essere ritrovato con un GET#. La sintassi di questo comando e':

```
M-R ,CHR$(LO), CHR$(HI)
```

Dove LO sta ad indicare il Byte basso dell' indirizzo nella dell' unita' a disco che deve essere letta ed HI il Byte alto.
Il seguente programma chiede un indirizzi e legge cio' che e' presente in quella locazione.

```
100 INPUT"INDIRIZZO ";A
110 HI=INT(A/256)
120 LO=A-256*HI
130 OPEN1,8,15
140 PRINT#1,"M-R";CHR$(LO);CHR$(HI)
150 GET#1,A$
160 PRINT ASC(A$+CHR$(0))
```

Per esempio se si vuol conoscere il numero di blocchi liberi su un dischetto non avremo la necessita' di leggere l'intera Directory, in quanto possono essere letti direttamente i Bytes appropriati dal SISTEMA OPERATIVO DISCO.
Cio' puo' rendersi necessario se non sappiamo se il file che vogliamo memorizzare ha abbastanza spazio su disco.

```
100 OPEN 1,8,15,"IO"
110 PRINT#1,"M-R";CHR$(250);CHR$(2)
120 GET#1,A$:IFA$=""THENA$=CHR$(0)
130 PRINT#1,"M-R";CHR$(252);CHR$(2)
140 GET#1,B$:IFB$=""THEN B$=CHR$(0)
150 PRINT ASC(A$)+256*ASC(B$) "BLOCKS FREE"
160 CLOSE 1
```

Con questa sintassi un comando MEMORY-READ dovra' essere dato per ogni Byte che si vuole leggere.
Tuttavia puo' rendersi necessario di leggere e di verificare piu' di un Byte per volta con un comando M-R.
Sara' quindi opportuno dare il numero di Bytes che devono essere letti con un terzo parametro:

```
M-R  CHR$(LO) CHR$(HI)  CHR$(NUM)
```

Si puo' utilizzare questo sistema per leggere il nome di un dischetto dal Buffer della BAM dove era in precedenza immagazzinato.
Prima di effettuare questa operazione il dischetto deve essere inizializzato in maniera tale che l' attuale numero del dischetto stesso sia immagazzinato nel Buffer di indirizzo $0700, dal quale sara' letto con un comando M-R.

```
100 OPEN 1,8,15,"IO"
110 PRINT#1,"M-R"CHR$(144)CHR$(7)CHR$(16)
120 INPUT#1,A$
130 PRINT A$
```

Questo e' un semplice sistema per leggere il nome del dischetto che ricordiamo e' costituito da 16 caratteri.
Con questa piccola routine da aggiungere in qualsiasi parte di un programma si puo' ad esempio controllare se il dischetto presente in quel momento nel drive e' quello giusto.

Anche il Buffer del disco puo' essere letto utilizzando questo metodo.
E' chiaro che cosi' si puo' anche leggere una parte del

DOS che e' su ROM copiandolo su un BUFFER, modificandola e quindi far girare la parte cambiata.

MEMORY-WRITE M-W

Il comando complementare del precedente e' il comando per scrivere dati all' interno della memoria RAM del drive. Naturalmente dove questo e' consentito cioe' nelle zone libere.
La sintassi del comando e' la seguente:

M-W CHR$(LO) CHR$(HI) CHR$(N) CHR$(DATA1) CHR$(DATA2)

Il numero di Bytes del parametro CHR$(N) cioe' il numero di Bytes che possono essere trasmessi e' teoricamente pari a 255.
Tuttavia a causa della limitazione del Buffer di INPUT (di ingresso dati) che puo' manipolare solo fino a 40 caratteri il numero di Bytes viene limitato a 34.
Un possibile uso di questo comando e' quello di cambiare il numero di device.
Il device e' immagazzinato in due locazioni di memoria in pagina zero.
Il numero della periferica piu' 32 ($20) e' immagazzinato nella locazione 177 ($77) chiamata anche LISTEN ADDRESS e che serve per ricevere dati dal computer.
Gli indirizzi immediatamente seguenti contengono il numero della periferica piu' il valore 64 ($40) per la funzione TALK che serve per inviare dati al computer.
Poiche' gli indirizzi sono salvati separatamente e'

possibile utilizzarne di differenti per l' invio e la ricezione di dati.
Nel seguente esempio l'indirizzo di ricezione e' fissato a 9,mentre l' indirizzo d' invio e' fissato sul valore 10.

```
100 OPEN1,8,15
110
PRINT#1,"M-W"CHR$(119)CHR$(0)CHR$(2)CHR$(9+32)CHR$(10+64
)
120 CLOSE 1
140 OPEN1,9,15
150 OPEN2,10,15
160 PRINT#1,"IO"
170 INPUT#2,A$,B$,C$,D$
180 PRINTA$","B$","C$","D$

        00,OK,00,00
```

E' necessario cambiare il numero di periferica se si desidera utilizzare piu' di un' unita' a dischi. Per questo cambiare il numero di periferica della seconda unita' a dischi in 9.
Questo cambiamento effettuato via SOFTWARE resta operante fino a quando non si proceda ad un RESET di sistema che potra' essere ottenuto sia sull' unita' a disco sia sull' unita' centrale.

NOTA

Se dovessero rendersi necessari cambiamenti permanenti, cioe' se si desidera utilizzare sempre due unita' a

dischi, si consiglia di far effettuare l' opportuna modifica hardware presso un qualsiasi centro di assistenza COMMODORE.

Poiche' molti parametri del DOS sono su RAM si possono eseguire cambiamenti su larga scala relativamente alle funzioni del SISTEMA OPERATIVO DISCO stesso come per esempio lo STEP SIZE, con il quale viene determinato il numero di settori per traccia.
L' indirizzo relativo a questa funzione e' il 105 ($69) che normalmente contiene il valore 10.
Si puo' anche dichiarare un numero di tentativi di lettura prima che venga restituito l'errore, cambiando il contenuto all' indirizzo 106($6A) che normalmente contiene il valore 5.

MEMORY-EXECUTE M-E

Utilizzando questo comando si puo' chiamare e mettere in funzione, cioe' far eseguire una routine in linguaggio macchina che sia presente nella memoria del SISTEMA OPERATIVO DISCO.
Come gli altri, anche questo comando puo' essere dato con una forma ridotta M-E, tenendo tuttavia presente che la routine che andra' in esecuzione deve terminare con un RTS (RETURN FROM SUBROUTINE) che ha per codice $60.
La sintassi del comando e':

```
M-E  CHR$(LO)  CHR$(HI)
```

Come sempre i due valori dati dai parametri CHR$ stanno ad indicare i bytes relativi all' indirizzo di partenza

della subroutine in linguaggio macchina che questo comando serve a mettere in funzione.
Sara' quindi possibile chiamare dal SISTEMA OPERATIVO DISCO una subroutine come e' stato fatto relativamente al comando M-W, e poi eseguirla.
Per esempio si puo' chiamare una routine che crea un messaggio d'errore. L' indirizzo $EFC9 e' il punto di partenza della routine che visualizza il messaggio:

72,disk full

```
100 OPEN 1,8,15
110 PRINT#1,"M-E"CHR$(201)CHR$(239)
120 INPUT#1,A$,B$,C$,D$
130 PRINTA$","B$","C$","D$
```

Nella linea 110 l' indirizzo $EFC9 e' scomposto nel byte basso ($C9) e byte alto ($EF) ed inviato come parametro del comando M-E. Viene quindi letto il canale d'errore ed il seguente messaggio visualizzato:

72 DISK FULL,00,00

Se desiderate far girare un vostro programma sul 1541, questi dovrebbe essere scritto su un Buffer libero e quindi richiamato con un comando M-E.
Dovendo poi utilizzare spesso questo programma, il contenuto del buffer dovrebbe essere scritto su un blocco del dischetto.
Potra' essere quindi eseguito con un comando B-E che carichera' il contenuto del blocco in un buffer e fara' partire automaticamente la routine.

I COMANDI USER U

Utilizzando questi tipi di comandi, cioe' i comandi USER (U) ci sono due strade per eseguire i programmi su disco.
La sintassi di questi comandi e' la seguente:

UX

Dove X puo' essere una lettera da A a J o un numero da 1 a 9 o un (:) che prende il posto di 10.
Quando un comando di questo tipo e' messo in funzione, viene eseguito un salto ad uno dei seguenti indirizzi del SISTEMA OPERATIVO DISCO:

```
UA  U1  $CD5F
UB  U2  $DC97
UC  U3  $0500
UD  U4  $0503
UE  U5  $0506
UF  U6  $0509
UG  U7  $050C
UH  U8  $050F
UI  U9  $FF01
UJ  U:  $EAA0
```

Ricordiamo che i comandi U1 e U2 (oppure gli equivalenti UA e UB) ppossono essere utilizzati al posto di BLOCK-READ e BLOCK-WRITE.
I comandi da U3 a U8 saltano a degli indirizzi nel Buffer numero 2 di indirizzo $500.
Se si desiderano utilizzare numerosi comandi allora deve essere proprio in questa zona che sono immessi i

valori di salto alle singole routines.
Se invece viene utilizzato un singolo comando U3 il programma puo' iniziare direttamente da $500.

Il comando UJ salta al cosi' detto VETTORE DI RESET e di conseguenza l' unita' a dischi viene resettata.

```
100 OPEN 1,8,15
110 PRINT#1,"UJ"
120 FORI=1 TO 1000:NEXT
130 GET#1,A$:PRINTA$:IFST¨è64THEN130
```

```
        73,CBM DOS V2.6 1541,00,00
```

Utilizzando i comandi USER i parametri possono essere passati alle routines.
Possibili parametri possono essere indirizzi, codici di comando e nomi di files.
Questo sistema puo' essere utilizzato per espandere i comandi del disco o per realizzare nuove strutture di dati.

OPERAZIONI DEL DOS

Il 1541, seguendo una costante nella politica della COMMODORE e' un' unita' a dischi intelligente con un suo microprocessore ed un suo SISTEMA OPERATIVO appunto il DOS o DISK OPERATING SYSTEM.
Cio' naturalmente significa che non e' necessario nesun spazio nella memoria centrale del computer e che quindi

e' sufficiente trasmettere i dati al disco senza nessuna sequenza speciale di temporizzazione.
L' unita' a dischi esegue quindi tre operazioni simultaneamente.
Per primo controlla il traffico dei dati fra esso stesso e l' unita' centrale.
Secondariamente interpreta i comandi ed esegue operazioni a livello di manipolazione dei files, dei relativi canali di comunicazione associati e dei buffers.
Terzo esegue una serie di operazioni chiamate HARDWARE-ORIENTED come la formattazione del dischetto, lettura e scrittura dati e programmi, cancellazione, validate, ecc.

Queste operazioni sono messe in funzione simultaneamente dal microprocessore 6502 attraverso l' utilizo delle tecniche di INTERRUPT.

Molte delle funzioni relative al DOS riguardano l'esecuzione dei comandi trasmessi dal computer.
La ricezione dei dati e comandi dal computer e' controllata dagli INTERRUPTS.
Se il computer desidera colloquiare con un' unita' periferica dovra' inviare un impulso lungo la linea di ATN.
Cio' genera un' interruzione (appunto un INTERRUPT) sull' unita' a dischi.
A questo punto il DOS interrompe il lavoro che attualmente sta eseguendo e prende nota che il computer desidera inviare dei dati.
E' da notare il termine prende nota perche' in effetti il DOS riprende immediatamnte il suo lavoro, lo porta a termine e si prepara a ricevere i dati dal computer..
Quando l' invio dei comandi e' terminato, il disco resta in posizione o ciclo d' attesa per nuovi comandi o dati.

L' esecuzione di un comando a questo livello e' limitata all' esecuzione logica del comando, al controllo dei canali di comunicazione da e per il computer e la preparazione e la ricerca di dati che devono rispettivamente essere letti o scritti.
Le operazioni· del controller, di formattazione, scrittura e lettura dei singoli blocchi deve essere eseguita dal processore.
Anche queste operazioni sono controllate dalle fasi di INTERRUPT.
L' esecuzione dei normali programmi e' interrotta ogni 14 millisecondi da un temporizzatore.
La comunicazione fra due programmi indipendenti e' manipolata attraverso una comune area di memoria.

Informazioni piu' dettagliate ed approfondite circa il modo ed i sistemi di lavoro del disco si possono trovare nel gia' citato manuale LE PERIFERICHE COMMODORE ed. EVM

A partire dalle seguenti pagine riportiamo il contenuto delle prime pagine di memoria del disco e l' intero DOS disassemblato.

LE PORTE DI INGRESSO E USCITA

VIA 6522 1, Porta per il BUS seriale

```
$1800   Port  B
$1801   Port  A
$1802   DATA DIRECTION  B
$1803   DATA DIRECTION  A
PB 0  :   DATA IN
PB 1  :   DATA OUT
PB 2  :   CLOCK IN
PB 3  :   CLOCK OUT
PB 4  :   ATN A
PB 5,6:   Indirizzo periferica
CB 2  :   ATN IN
```

VIA 6522 2, Porta motore e R/W HEAD CONTROL

```
$1C00   Porta B, controllo di porta
$1C01   Porta A, dati a e da R/W HEAD
$1C02   DATA DIRECTION  A
$1C03   DATA DIRECTION  B
PB 0:   STP I
PB 1:   STP 0 STEPPER MOTOR(movimento
        testina)
PB 2:   MTR pilotaggio motore
PB 3:   ACT accensione LED
PB 4:   WPS SWITCH di protezione scrittura.
PB 7:   SYNC
CA 1:   Byte ready
CA 2:   SOE
```

PRINCIPALI INDIRIZZI DI MEMORIA DISCO

0	$00	Codice di comando per Buffer 0
1	$01	Codice di comando per Buffer 1
2	$02	Codice di comando per Buffer 2
3	$03	Codice di comando per Buffer 3
4	$04	Codice di comando per Buffer 4
6	$06-07	Traccia e settore per Buffer 0
8	$08-09	Traccia e settore per Buffer 1
10	$0A-0B	Traccia e settore per Buffer 2
12	$0C-0D	Traccia e settore per Buffer 3
14	$0E-0F	Traccia e settore per Buffer 4
18	$12-13	ID per drive 0
20	$14-15	ID per drive 1
22	$16-17	ID
32	$20-21	FLAG per movimento testina
48	$30-31	Puntatore Buffer per Disk Controller
57	$39	Costante 8, segna l' inizio dei dati del blocco di testa
58	$3A	Segnale di parita' per i dati del Buffer
61	$3D	N. drive per il Disk Controller
63	$3F	N. Buffer per il Disk Controller
67	$43	N. dei settori per traccia per la formattazione
71	$47	Costante 7, segna l' inizio dei dati del blocco di testa
73	$49	Stack Pointer
74	$4A	Contatore passi per testina
81	$51	N. dell' attuale traccia per la formattazione.

105	$69	Ampiezza per la divisione di settore (10)
106	$6A	N. dei tentativi di lettura.
111	$6F-70	Puntatore all' indirizzo per comandi M e B.
119	$77	N. di periferica (+$20) per LISTEN
120	$78	N. di periferica (+$40) per TALK
121	$79	Flag per LISTEN
122	$7A	Flag per TALK
124	$7C	Flag per ATN di ricez. dati dal Bus seriale
125	$7D	Flag per EOI dal Bus seriale
127	$7F	N. di periferica.
128	$80	N. traccia
129	$81	N. settore
130	$82	N. canale
131	$83	Indirizzo secondario
132	$84	Indirizzo secondario
133	$85	Byte dati
139	$8B-8D	Area di immag. per divisione
148	$94-95	Attuale puntatore del Buffer
153	$99-9A	Indirizzo Buffer 0 ($300)
155	$9B-9C	Indirizzo Buffer 1 ($400)
157	$9D-9E	Indirizzo Buffer 2 ($500)
159	$9F-A0	Indirizzo Buffer 3 ($600)
161	$A1-A2	Indirizzo Buffer 4 ($700)
163	$A3-A4	Puntatore al Buffer di input a $200
165	$A5-A6	Puntatore al Buffer per i mess. di errore.
181	$B5-BA	Record LO, Blocco LO
187	$BB-C0	Record HI, Blocco HI.
193	$C1-C6	Puntatore scrittura file rel.
199	$C7-CC	Lunghezza record per file rel.
212	$D4	Puntatore NEL record per file rel.
213	$D5	N. di side sector
214	$D6	Puntatore al blocco dati nel side

		sector
215	$D7	Puntatore al record nel file rel.
231	$E7	Tipo di file
249	$F9	N. Buffer
256-325	$100-145	Area di stack
512-552	$200-228	Buffer per stringa comando
586	$24A	Tipo di file
600	$258	Lunghezza record
601	$259	Traccia side sector
602	$25A	Settore side sector
628	$274	Lunghezza della linea di input
632	$278	N. nomi di files
633	$279	Metodo controllo file
640-644	$280-284	Traccia di un file
645-649	$285-289	Settore di un file
725-761	$2D5-2F9	Buffer per mess. d' errore
762-764	$2FA-2FC	N. di blocchi liberi
768-1023	$300-3FF	Buffer 0
1024-1279	$400-4FF	Buffer 1
1280-1535	$500-5FF	Buffer 2
1536-1791	$600-6FF	Buffer 3
1792-2047	$700-7FF	Buffer 4

```
********************************    turn LED on
C100    78          SEI
C101    A9 F7       LDA #$F7        erase LED bit
C103    2D 00 1C    AND $1C00
C106    48          PHA
C107    A5 7F       LDA $7F         drive number
C109    F0 05       BEQ $C110       0?
C10B    68          PLA
C10C    09 00       ORA #$00        not drive 0, turn LED off
C10E    D0 03       BNE $C113
C110    68          PLA
C111    09 08       ORA #$08        turn LED on
C113    8D 00 1C    STA $1C00
C116    58          CLI
C117    60          RTS

********************************    turn LED on
C118    78          SEI
C119    A9 08       LDA #$08
C11B    0D 00 1C    ORA $1C00       LED on
C11E    8D 00 1C    STA $1C00
C121    58          CLI
C122    60          RTS

********************************    erase error flags
C123    A9 00       LDA #$00
C125    8D 6C 02    STA $026C
C128    8D 6D 02    STA $026D
C12B    60          RTS

********************************
C12C    78          SEI
C12D    8A          TXA             save X register
C12E    48          PHA
C12F    A9 50       LDA #$50
C131    8D 6C 02    STA $026C
C134    A2 00       LDX #$00
C136    BD CA FE    LDA $FECA,X     8
C139    8D 6D 02    STA $026D
C13C    0D 00 1C    ORA $1C00
C13F    8D 00 1C    STA $1C00       turn LED on
C142    68          PLA
C143    AA          TAX             get x register back
C144    58          CLI
C145    60          RTS

********************************    interpret command from
                                    computer
C146    A9 00       LDA #$00
C148    8D F9 02    STA $02F9
C14B    AD 8E 02    LDA $028E       last drive number
```

```
C14E   85 7F      STA $7F        drive number
C150   20 BC E6   JSR $E6BC      prepare 'ok' message
C153   A5 84      LDA $84        secondary address
CI55   10 09      BPL $C160
C157   29 0F      AND #$0F
C159   C9 0F      CMP #$0F       15, command channel
C15B   F0 03      BEQ $C160      yes
C15D   4C B4 D7   JMP $D7B4      to OPEN command
C160   20 B3 C2   JSR $C2B3      determine line length and
                                 erase flags
C163   B1 A3      LDA ($A3),Y    get first character
C165   8D 75 02   STA $0275      and store
C168   A2 0B      LDX #$0B       11
C16A   BD 89 FE   LDA $FE89,X    commands
C16D   CD 75 02   CMP $0275      compare to first character
C170   F0 08      BEQ $C17A      found?
C172   CA         DEX
C173   10 F5      BPL $C16A
C175   A9 31      LDA #$31       not found
C177   4C C8 C1   JMP $C1C8      31, 'syntax error'
C17A   8E 2A 02   STX $022A      number of command words
C17D   E0 09      CPX #$09
C17F   90 03      BCC $C184      command number < 9?
C181   20 EE C1   JSR $C1EE      test for 'R', 'S', and 'N'
C184   AE 2A 02   LDX $022A      command number
C187   BD 95 FE   LDA $FE95,X    jump address lo
C18A   85 6F      STA $6F
C18C   BD A1 FE   LDA $FEA1,X    jump address hi
C18F   85 70      STA $70
C191   6C 6F 00   JMP ($006F)    jump to command

*******************************  prepare error message after
                                 executing command
C194   A9 00      LDA #$00
C196   8D F9 02   STA $02F9
C199   AD 6C 02   LDA $026C      flag set?
C19C   D0 2A      BNE $C1C8      yes, then set error message
C19E   A0 00      LDY #$00
C1A0   98         TYA            error number 0
C1A1   84 80      STY $80        track number 0
C1A3   84 81      STY $81        sector number 0
C1A5   84 A3      STY $A3
C1A7   20 C7 E6   JSR $E6C7      prepare 'ok' message
C1AA   20 23 C1   JSR $C123      erase error flag
C1AD   A5 7F      LDA $7F        drive number
C1AF   8D 8E 02   STA $028E      save as last drive number
C1B2   AA         TAX
C1B3   A9 00      LDA #$00
C1B5   95 FF      STA $FF,X
C1B7   20 BD C1   JSR $C1BD      erase input buffer
C1BA   4C DA D4   JMO $D4DA      close internal channel

*******************************  erase input buffer
C1BD   A0 28      LDY #$28       erase 41 characters
C1BF   A9 00      LDA #$00
```

```
C1C1   99 00 02   STA $0200,Y    $200 to $228
C1C4   88         DEY
C1C5   10 FA      BPL $C1C1
C1C7   60         RTS
******************************   give error message
                                 (track & sector)
C1C8   A0 00      LDY #$00
C1CA   84 80      STY $80        track = 0
C1CC   84 81      STY $81        sector = 0
C1CE   4C 45 E6   JMP $E645      error number acc, generate
                                 error message

******************************   check input line
C1D1   A2 00      LDX #$00
C1D3   8E 7A 02   STX $027A      pointer to drive number
C1D6   A9 3A      LDA #$3A       ':'
C1D8   20 68 C2   JSR $C268      test line to ':' or to end
C1DB   F0 05      BEQ $C1E2      no colon found?
C1DD   88         DEY
C1DE   88         DEY
C1DF   8C 7A 02   STY $027A      point to drive number
                                 (before colon)
C1E2   4C 68 C3   JMP $C368      get drive # and turn LED on

******************************   check input line
C1E5   A0 00      LDY #$00       pointer to input buffer
C1E7   A2 00      LDX #$00       counter for commas
C1E9   A9 3A      LDA #$3A       ':'
C1EB   4C 68 C2   JMP $C268      test line to colon or to end

******************************   check input line
C1EE   20 E5 C1   JSR $C1E5      test line to ':' or end
C1F1   D0 05      BNE $C1F8      colon found?
C1F3   A9 34      LDA #$34
C1F5   4C C8 C1   JMP $C1C8      34, 'syntax error'
C1F8   88         DEY
C1F9   88         DEY            set pointer to colon
C1FA   8C 7A 02   STY $027A      position of the drive no.
C1FD   8A         TXA            comma before the colon
C1FE   D0 F3      BNE $C1F3      yes, then 'syntax error'
C200   A9 3D      LDA #$3D       '='
C202   20 68 C2   JSR $C268      check input to '='
C205   8A         TXA            comma found?
C206   F0 02      BEQ $C20A      no
C208   A9 40      LDA #$40       bit 6
C20A   09 21      ORA #$21       and set bit 0 and 5
C20C   8D 8B 02   STA $028B      flag for syntax check
C20F   E8         INX
C210   8E 77 02   STX $0277
C213   8E 78 02   STX $0278
C216   AD 8A 02   LDA $028A      wildcard found?
C219   F0 0D      BEQ $C228      no
C21B   A9 80      LDA #$80
C21D   0D 8B 02   ORA $028B      set bit 7
C220   8D 8B 02   STA $028B
```

```
C223    A9 00       LDA #$00
C225    8D 8A 02    STA $028A      reset wildcard flag
C228    98          TYA            '=' found?
C229    F0 29       BEQ $C254      no
C22B    9D 7A 02    STA $027A,X
C22E    AD 77 02    LDA $0277      number of commas before '='
C231    8D 79 02    STA $0279
C234    A9 8D       LDA #$8D       shift CR
C236    20 68 C2    JSR $C268      check line to end
C239    E8          INX            increment comma counter
C23A    8E 78 02    STX $0278      store # of commas
C23D    CA          DEX
C23E    AD 8A 02    LDA $028A      wildcard found?
C24A    F0 02       BEQ $C245      no
C243    A9 08       LDA #$08       set bit 3
C245    EC 77 02    CPX $0277      comma after '='?
C248    F0 02       BEQ $C24C      no
C24A    09 04       ORA #$04       set bit 2
C24C    09 03       ORA #$03       set bits 0 and 1
C24E    4D 8B 02    EOR $028B
C251    8D 8B 02    STA $028B      as flag for syntax check
C254    AD 8B 02    LDA $028B      syntax flag
C257    AE 2A 02    LDX $022A      command number
C25A    3D A5 FE    AND $FEA5,X    combine with check byte
C25D    D0 01       BNE $C260
C25F    60          RTS
C260    8D 6C 02    STA $026C      set error flag
C263    A9 30       LDA #$30
C265    4C C8 C1    JMP $C1C8      30, 'syntax error'

******************************     search characters in input
                                   buffer
C268    8D 75 02    STA $0275      save character
C26B    CC 74 02    CPY $0274      already done?
C26E    B0 2E       BCS $C29E      yes
C270    B1 A3       LDA ($A3),Y    get char from buffer
C272    C8          INY
C273    CD 75 02    CMP $0275      compared with char
C276    F0 28       BEQ $C2A0      found
C278    C9 2A       CMP #$2A       '*'
C27A    F0 04       BEQ $C280
C27C    C9 3F       CMP #$3F       '?'
C27E    D0 03       BNE $C283
C280    EE 8A 02    INC $028A      set wildcard flag
C283    C9 2C       CMP #$2C       ','
C285    D0 E4       BNE $C26B
C287    98          TYA
C288    9D 7B 02    STA $027B,X    note comma position
C28B    AD 8A 02    LDA $028A      wildcard flag
C28E    29 7F       AND #$7F
C290    F0 07       BEQ $C299      no wildcard
C292    A9 80       LDA #$80
C294    95 E7       STA $E7,X      note flag
C296    8D 8A 02    STA $028A      and save as wildcard flag
C299    E8          INX            inc comma counter
```

```
C29A    E0 04       CPX #$04       4 commas already?
C29C    90 CD       BCC $C26B      no, continue
C29E    A0 00       LDY #$00
C2A0    AD 74 02    LDA $0274      set flag for line end
C2A3    9D 7B 02    STA $027B,X
C2A6    AD 8A 02    LDA $028A      wildcard flag
C2A9    29 7F       AND #$7F
C2AB    F0 04       BEQ $C2B1      no wildcard
C2AD    A9 80       LDA #$80
C2AF    95 E7       STA $E7,X      set flag
C2B1    98          TYA
C2B2    60          RTS

******************************     check line length
C2B3    A4 A3       LDY $A3        ptr to command input buffer
C2B5    F0 14       BEQ $C2CB      zero?
C2B7    88          DEY
C2B8    F0 10       BEQ $C2CA      one?
C2BA    B9 00 02    LDA $0200,Y    pointer to input buffer
C2BD    C9 0D       CMP #$0D       'CR'
C2BF    F0 0A       BEQ $C2CB      yes, line end
C2C1    88          DEY
C2C2    B9 00 02    LDA $0200,Y    preceding character
C2C5    C9 0D       CMP #$0D       'CR'
C2C7    F0 02       BEQ $C2CB      yes
C2C9    C8          INY
C2CA    C8          INY            pointer to old value again
C2CB    8C 74 02    STY $0274      same line length
C2CE    C0 2A       CPY #$2A       compare with 42 characters
C2D0    A0 FF       LDY #$FF
C2D2    90 08       BCC $C2DC      smaller, ok
C2D4    8C 2A 02    STY $022A
C2D7    A9 32       LDA #$32
C2D9    4C C8 C1    JMP $C1C8      32, 'syntax error' line too
                                   long

******************************     erase flag for input command
C2DC    A0 00       LDY #$00
C2DE    98          TYA
C2DF    85 A3       STA $A3        pointer to input buffer lo
C2E1    8D 58 02    STA $0258      record length
C2E4    8D 4A 02    STA $024A      file type
C2E7    8D 96 02    STA $0296
C2EA    85 D3       STA $D3
C2EC    8D 79 02    STA $0279      comma counter
C2EF    8D 77 02    STA $0277           "
C2F2    8D 78 02    STA $0278           "
C2F5    8D 8A 02    STA $028A      wildcard flag
C2F8    8D 6C 02    STA $026C      error flag
C2FB    A2 05       LDX #$05
C2FD    9D 79 02    STA $0279,X    flags for line analysis
C300    95 D7       STA $D7,X      directory sectors
C302    95 DC       STA $DC,X      buffer pointer
C304    95 E1       STA $E1,X      drive number
C306    95 E6       STA $E6,X      wildcard flag
```

```
C308    9D 7F 02    STA $027F,X    track number
C30B    9D 84 02    STA $0284,X    sector number
C30E    CA          DEX
C30F    D0 EC       BNE $C2FD
C311    60          RTS

******************************     preserve drive number
C312    AD 78 02    LDA $0278      number of commas
C315    8D 77 02    STA $0277      save
C318    A9 01       LDA #$01
C31A    8D 78 02    STA $0278      number of drive numbers
C31D    8D 79 02    STA $0279
C320    AC 8E 02    LDY $028E      last drive number
C323    A2 00       LDX #$00
C325    86 D3       STX $D3
C327    BD 7A 02    LDA $027A,X    position of the colon
C32A    20 3C C3    JSR $C33C      get drive no. before colon
C32D    A6 D3       LDX $D3
C32F    9D 7A 02    STA $027A      save exact position
C332    98          TYA
C333    95 E2       STA $E2,X      drive number in table
C335    E8          INX
C336    EC 78 02    CPX $0278      got all drive numbers?
C339    90 EA       BCC $C325      no, continue
C33B    60          RTS

******************************     search for drive number
C33C    AA          TAX            note position
C33D    A0 00       LDY #$00
C33F    A9 3A       LDA #$3A       ':'
C341    DD 01 02    CMP $0201,X    colon behind it?
C344    F0 0C       BEQ $C352      yes
C346    DD 00 02    CMP $0200,X    colon here?
C349    D0 16       BNE $C361      no
C34B    E8          INX
C34C    98          TYA
C34D    29 01       AND #$01       drive number
C34F    A8          TAY
C350    8A          TXA
C351    60          RTS

C352    BD 00 02    LDA $0200,X    get drive number
C355    E8          INX
C356    E8          INX
C357    C9 30       CMP #$30       '0'?
C359    F0 F2       BEQ $C34D      yes
C35B    C9 31       CMP #$31       '1'?
C35D    F0 EE       BEQ $C34D      yes
C35F    D0 EB       BNE $C34C      no, use last drive number
C361    98          TYA            last drive number
C362    09 80       ORA #$80       set bit 7, uncertain drive #
C364    29 81       AND #$81       erase remaining bits
C366    D0 E7       BNE $C34F

******************************     get drive number
```

```
C368    A9 00       LDA #$00
C36A    8D 8B 02    STA $028B       erase syntax flag
C36D    AC 7A 02    LDY $027A       position in command line
C370    B1 A3       LDA ($A3),Y     get chars from command buffer
C372    20 BD C3    JSR $C3BD       get drive number
C375    10 11       BPL $C388       certain number?
C377    C8          INY             increment pointer
C378    CC 74 02    CPY $0274       line end?
C37B    B0 06       BCS $C383       yes
C37D    AC 74 02    LDY $0274
C380    88          DEY
C381    D0 ED       BNE $C370       search line for drive no.
C383    CE 8B 02    DEC $028B
C386    A9 00       LDA #$00
C388    29 01       AND #$01
C38A    85 7F       STA $7F         drive number
C38C    4C 00 C1    JMP $C100       turn LED on

******************************      reverse drive number
C38F    A5 7F       LDA $7F         drive number
C391    49 01       EOR #$01        switch bit 0
C393    29 01       AND #$01
C395    85 7F       STA $7F
C397    60          RTS

******************************      establish file type
C398    A0 00       LDY #$00
C39A    AD 77 02    LDA $0277       '=' found?
C39D    CD 78 02    CMP $0278
C3A0    F0 16       BEQ $C3B8       no
C3A2    CE 78 02    DEC $0278       get pointer
C3A5    AC 78 02    LDY $0278
C3A8    B9 7A 02    LDA $027A,Y     set pointer to character
                                    behind '='
C3AB    A8          TAY
C3AC    B1 A3       LDA ($A3),Y     pointer to buffer
C3AE    A0 04       LDY #$04        compare with marker for
                                    file type
C3B0    D9 BB FE    CMP $FEBB,Y     'S', 'P', 'U', 'R'
C3B3    F0 03       BEQ $C3B8       agreement
C3B5    88          DEY
C3B6    D0 F8       BNE $C3B0
C3B8    98          TYA
C3B9    8D 96 02    STA $0296       note file type (1-4)
C3BC    60          RTS

******************************      check drive number
C3BD    C9 30       CMP #$30        '0'
C3BF    F0 06       BEQ $C3C7
C3C1    C9 31       CMP #$31        '1'
C3C3    F0 02       BEQ $C3C7
C3C5    09 80       ORA #$80        no zero or one, then set bit 7
C3C7    29 81       AND #$81
C3C9    60          RTS
```

```
********************************   verify drive number
C3CA   A9 00      LDA #$00
C3CC   85 6F      STA $6F
C3CE   8D 8D 02   STA $028D
C3D1   48         PHA
C3D2   AE 78 02   LDX $0278      number of drive numbers
C3D5   68         PLA
C3D6   05 6F      ORA $6F
C3D8   48         PHA
C3D9   A9 01      LDA #$01
C3DB   85 6F      STA $6F
C3DD   CA         DEX
C3DE   30 0F      BMI $C3EF
C3E0   B5 E2      LDA $E2,X
C3E2   10 04      BPL $C3E8
C3E4   06 6F      ASL $6F
C3E6   06 6F      ASL $6F
C3E8   4A         LSR A
C3E9   90 EA      BCC $C3D5
C3EB   06 6F      ASL $6F
C3ED   D0 E6      BNE $C3D5
C3EF   68         PLA
C3F0   AA         TAX
C3F1   BD 3F C4   LDA $C43F,X    get syntax flag
C3F4   48         PHA
C3F5   29 03      AND #$03
C3F7   8D 8C 02   STA $028C
C3FA   68         PLA
C3FB   0A         ASL A
C3FC   10 3E      BPL $C43C
C3FE   A5 E2      LDA $E2
C400   29 01      AND #$01       isolate drive number
C402   85 7F      STA $7F
C404   AD 8C 02   LDA $028C
C407   F0 28      BEQ $C434
C409   20 3D C6   JSR $C63D      initialze drive
C40C   F0 12      BEQ $C420      error?
C40E   20 8F C3   JSR $C38F      switch to other drive
C411   A9 00      LDA #$00
C413   8D 8C 02   STA $028C
C416   20 3D C6   JSR $C63D      initialize drive
C419   F0 1E      BEQ $C439      no error?
C41B   A9 74      LDA #$74
C41D   20 C8 C1   JSR $C1C8      74, 'drive not ready'
C420   20 8F C3   JSR $C38F

C423   20 3D C6   JSR $C63D      initialize drive
C426   08         PHP
C427   20 8F C3   JSR $C38F      switch to other drive
C42A   28         PLP
C42B   F0 0C      BEQ $C439      no error?
C42D   A9 00      LDA #$00
C42F   8D 8C 02   STA $028C      number of drives
C432   F0 05      BEQ $C439
C434   20 3D C6   JSR $C63D      initialize drive
```

```
C437    D0 E2      BNE $C41B      error?
C439    4C 00 C1   JMP $C100      Turn LED on
C43C    2A         ROL A          drive # from carry after bit 0
C43D    4C 00 C4   JMP $C400

******************************    flags for drive check
C440    00 80 41 01 01 01 01 81
C448    81 81 81 42 42 42 42

******************************    search for file in directory
C44F    20 CA C3   JSR $C3CA      initialize drive
C452    A9 00      LDA #$00
C454    8D 92 02   STA $0292      pointer
C457    20 AC C5   JSR $C5AC     read first directory block
C45A    D0 19      BNE $C475      entry present?
C45C    CE 8C 02   DEC $028C      drive number clear?
C45F    10 01      BPL $C462      no
C461    60         RTS

C462    A9 01      LDA #$01
C464    8D 8D 02   STA $028D
C467    20 8F C3   JSR $C38F      change drive
C46A    20 00 C1   JSR $C100      Turn LED on
C46D    4C 52 C4   JMP $C452      and search

C470    20 17 C6   JSR $C617      search next file in directory
C473    F0 10      BEQ $C485      not found?
C475    20 D8 C4   JSR $C4D8      verify directory entry
C478    AD 8F 02   LDA $028F
C47B    F0 01      BEQ $C47E      more files?
C47D    60         RTS

C47E    AD 53 02   LDA $0253
C481    30 ED      BMI $C470      file not found?
C483    10 F0      BPL $C475      yes
C485    AD 8F 02   LDA $028F
C488    F0 D2      BEQ $C45C
C48A    60         RTS

C48B    20 04 C6   JSR $C604      search next directory block
C48E    F0 1A      BEQ $C4AA      not found?
C490    D0 28      BNE $C4BA

C492    A9 01      LDA #$01
C494    8D 8D 02   STA $028D
C497    20 8F C3   JSR $C38F      change drive
C49A    20 00 C1   JSR $C100      turn LED on
C49D    A9 00      LDA #$00
C49F    8D 92 02   STA $0292
C4A2    20 AC C5   JSR $C5AC      read directory block
C4A5    D0 13      BNE $C4BA      found?
C4A7    8D 8F 02   STA $028F
C4AA    AD 8F 02   LDA $028F
C4AD    D0 28      BNE $C4D7
C4AF    CE 8C 02   DEC $028C
```

```
C4B2    10 DE       BPL $C492
C4B4    60          RTS

C4B5    20 17 C6    JSR $C617       next entry in directory
C4B8    F0 F0       BEQ $C4AA       not found?
C4BA    20 D8 C4    JSR $C4D8       check entry
C4BD    AE 53 02    LDX $0253
C4C0    10 07       BPL $C4C9       file found?
C4C2    AD 8F 02    LDA $028F
C4C5    F0 EE       BEQ $C4B5       yes
C4C7    D0 0E       BNE $C4D7       no, then done

C4C9    AD 96 02    LDA $0296
C4CC    F0 09       BEQ $C467
C4CE    B5 E7       LDA $E7,X       file type
C4D0    29 07       AND #$07
C4D2    CD 96 02    CMP $0296       same as desired file type?
C4D5    D0 DE       BNE $C4B5       no
C4D7    60          RTS

C4D8    A2 FF       LDX #$FF
C4DA    8E 53 02    STX $0253       flag for data found
C4DD    E8          INX
C4DE    8E 8A 02    STX $028A
C4E1    20 89 C5    JSR $C589       set pointer to data
C4E4    F0 06       BEQ $C4EC
C4E6    60          RTS

C4E7    20 94 C5    JSR $C594       pointer to next file
C4EA    D0 FA       BNE $C4E6       end, then done
C4EC    A5 7F       LDA $7F         drive number
C4EE    55 E2       EOR $E2,X
C4F0    4A          LSR A
C4F1    90 0B       BCC $C4FE
C4F3    29 40       AND #$40
C4F5    F0 F0       BEQ $C4E7
C4F7    A9 02       LDA #$02
C4F9    CD 8C 02    CMP $028C       search both drives?
C4FC    F0 E9       BEQ $C4E7       yes
C4FE    BD 7A 02    LDA $027A,X
C501    AA          TAX
C502    20 A6 C6    JSR $C6A6       get length of filename
C505    A0 03       LDY #$03
C507    4C 1D C5    JMP $C51D

C50A    BD 00 02    LDA $0200,X     get chars out of command line
C50D    D1 94       CMP ($94),Y     same character in directory?
C50F    F0 0A       BEQ $C51B       yes
C511    C9 3F       CMP #$3F        '?'
C513    D0 D2       BNE $C4E7       no
C515    B1 94       LDA ($94),Y
C517    C9 A0       CMP #$A0        shift blank, end of name?
C519    F0 CC       BEQ $C4E7       yes
C51B    E8          INX             increment pointer
C51C    C8          INY
```

```
C51D    EC 76 02    CPX $0276       end of the name in the command?
C520    B0 09       BCS $C52B       yes
C522    BD 00 02    LDA $0200,X     next character
C525    C9 2A       CMP #$2A        '*'
C527    F0 0C       BEQ $C535       yes, file found
C529    D0 DF       BNE $C50A       continue search

C52B    C0 13       CPY #$13        19
C52D    B0 06       BCS $C535       reached end of name
C52F    B1 94       LDA ($94),Y
C531    C9 A0       CMP #$A0        shift blank, end of name
C533    D0 B2       BNE $C4E7       not found
C535    AE 79 02    LDX $0279
C538    8E 53 02    STX $0253
C53B    B5 E7       LDA $E7,X
C53D    29 80       AND #$80
C53F    8D 8A 02    STA $028A
C542    AD 94 02    LDA $0294
C545    95 DD       STA $DD,X
C547    A5 81       LDA $81         sector number of the directory
C549    95 D8       STA $D8,X       enter in table
C54B    A0 00       LDY #$00
C54D    B1 94       LDA ($94),Y     file type
C54F    C8          INY
C550    48          PHA
C551    29 40       AND #$40        isolate scratch-protect bit
C553    85 6F       STA $6F         (6) and save
C555    68          PLA
C556    29 DF       AND #$DF        erase bit 7
C558    30 02       BMI $C55C
C55A    09 20       ORA #$20        set bit 5
C55C    29 27       AND #$27        erase bits 3 and 4
C55E    05 6F       ORA $6F         get bit 6 again
C560    85 6F       STA $6F
C562    A9 80       LDA #$80
C564    35 E7       AND $E7,X       isolate flag for wildcard
C566    05 6F       ORA $6F,X
C568    95 E7       STA $E7,X       write in table
C56A    B5 E2       LDA $E2,X
C56C    29 80       AND #$80
C56E    05 7F       ORA $7F         drive number
C570    95 E2       STA $E2,X
C572    B1 94       LDA ($94),Y
C574    9D 80 02    STA $0280,X     first track of file
C577    C8          INY
C578    B1 94       LDA ($94),Y
C57A    9D 85 02    STA $0285,X     get sector from directory
C57D    AD 58 02    LDA $0258       record length
C580    D0 07       BNE $C589
C582    A0 15       LDY #$15
C584    B1 94       LDA ($94),Y     record length
C586    8D 58 02    STA $0258       get from directory
C589    A9 FF       LDA #$FF
C58B    8D 8F 02    STA $028F
C58E    AD 78 02    LDA $0278
```

```
C591    8D 79 02    STA $0279
C594    CE 79 02    DEC $0279
C597    10 01       BPL $C59A
C599    60          RTS

C59A    AE 79 02    LDX $0279
C59D    B5 E7       LDA $E7,X      wildcard flag set?
C59F    30 05       BMI $C5A6      yes
C5A1    BD 80 02    LDA $0280,X    track number already set
C5A4    D0 EE       BNE $C594      yes
C5A6    A9 00       LDA #$00
C5A8    8D 8F 02    STA $028F
C5AB    60          RTS

C5AC    A0 00       LDY #$00
C5AE    8C 91 02    STY $0291
C5B1    88          DEY
C5B2    8C 53 02    STY $0253
C5B5    AD 85 FE    LDA $FE85      18, directory track
C5B8    85 80       STA $80
C5BA    A9 01       LDA #$01
C5BC    85 81       STA $81        sector 1
C4BE    8D 93 02    STA $0293
C5C1    20 75 D4    JSR $D475      read sector
C5C4    AD 93 02    LDA $0293
C5C7    D0 01       BNE $C5CA
C5C9    60          RTS

C5CA    A9 07       LDA #$07
C5CC    8D 95 02    STA $0295      number of directory entries (-1)
C5CF    A9 00       LDA #$00
C5D1    20 F6 D4    JSR $D4F6      get pointer from buffer
C5D4    8D 93 02    STA $0293      save as track number
C5D7    20 E8 D4    JSR $D4E8      set buffer pointer
C5DA    CE 95 02    DEC $0295      decrement counter
C5DD    A0 00       LDY #$00
C5DF    B1 94       LDA ($94),Y    first byte from directory
C5E1    D0 18       BNE $C5FB
C5E3    AD 91 02    LDA $0291
C5E6    D0 2F       BNE $C617
C5E8    20 3B DE    JSR $DE3B      get track and sector number
C5EB    A5 81       LDA $81
C5ED    8D 91 02    STA $0291      sector number
C5F0    A5 94       LDA $94
C5F2    AE 92 02    LDX $0292
C5F5    8D 92 02    STA $0292      buffer pointer
C5F8    F0 1D       BEQ $C617
C5FA    60          RTS

C5FB    A2 01       LDX #$01
C5FD    EC 92 02    CPX $0292      buffer pointer to one?
C600    D0 2D       BNE $C62F
C602    F0 13       BEQ $C617

C604    AD 85 FE    LDA $FE85      18, track number of BAM
```

```
C607    85 80       STA $80        track number
C609    AD 90 02    LDA $0290
C60C    85 81       STA $81        sector number
C60E    20 75 D4    JSR $D475      read block
C611    AD 94 02    LDA $0294
C614    20 C8 D4    JSR $D4C8      set buffer pointer
C617    AD FF       LDA #$FF
C619    8D 53 02    STA $0253      erase-file found flag
C61C    AD 95 02    LDA $0295
C61F    30 08       BMI $C629      all directory entries checked?
C621    A9 20       LDA #$20
C623    20 C6 D1    JSR $D1C6      inc buffer ptr by 32, next entry
C626    4C D7 C5    JMP $C567      and continue

C629    20 4D D4    JSR $D44D      set buffer pointer
C62C    4C C4 C5    JMP $C5C4      read next block

C62F    A5 94       LDA $94
C631    8D 94 02    STA $0294
C634    20 3B DE    JSR $DE3B      get track & sector no. from buffer
C637    A5 81       LDA $81
C639    8D 90 02    STA $0290      save sector number
C63C    60          RTS

********************************   test and initialize drive
C63D    A5 68       LDA $68
C63F    D0 28       BNE $C669
C641    A6 7F       LDX $7F        drive number
C643    56 1C       LSR $1C,X      disk changed?
C645    90 22       BCC $C669      no, then done
C647    A9 FF       LDA $FF
C649    8D 98 02    STA $0298      set error flag
C64C    20 0E D0    JSR $D00E      read directory track
C64F    A0 FF       LDY #$FF
C651    C9 02       CMP #$02       20, 'read error'?
C653    F0 0A       BEQ $C65F      yes
C655    C9 03       CMP #$03       21, 'read error'?
C657    F0 06       BEQ $C65F      yes
C659    C9 0F       CMP #$0F       74, 'drive not ready'?
C65B    F0 02       BEQ $C65F      yes
C65D    A0 00       LDY #$00
C65F    A6 7F       LDX $7F        drive number
C661    98          TYA
C662    95 FF       STA $FF,X      save error flag
C664    D0 03       BNE $C669      error?
C666    20 42 D0    JSR $D042      load BAM
C669    A6 7F       LDX $7F        drive number
C66B    B5 FF       LDA $FF,X      transmit error code
C66D    60          RTS

********************************   name of file in directory buffer
C66E    48          PHA
C66F    20 A6 C6    JSR $C6A6      get end of the name
C672    20 88 C6    JSR $C688      write filename in buffer
C675    68          PLA
```

```
C676    38          SEC
C677    ED 4B 02    SBC $024B       compare len with max length
C67A    AA          TAX
C67B    F0 0A       BEQ $C687
C67D    90 08       BCC $C687
C67F    A9 A0       LDA #$A0        pad with 'Shift blank'
C681    91 94       STA ($94),Y
C683    C8          INY
C684    CA          DEX
C685    D0 FA       BNE $C681
C687    60          RTS

*******************************
C688    98          TYA             buffer number
C689    0A          ASL A
C68A    A8          TAY             times 2 as pointer
C68B    B9 99 00    LDA $0099,Y
C68E    85 94       STA $94
C690    B9 9A 00    LDA $009A       buffer pointer after $94/$95
C693    85 95       STA $95
C695    A0 00       LDY #$00
C697    BD 00 02    LDA $0200,X     transmit characters in buffer
C69A    91 94       STA ($94),Y
C69C    C8          INY
C69D    F0 06       BEQ $C6A5       buffer already full?
C69F    E8          INX
C6A0    EC 76 02    CPX $0276
C6A3    90 F2       BCC $C697
C6A5    60          RTS

*******************************     search for end of name in command
C6A6    A9 00       LDA #$00
C6A8    8D 4B 02    STA $024B
C6AB    8A          TXA
C6AC    48          PHA
C6AD    BD 00 02    LDA $0200,X     get characters out of buffer
C6B0    C9 2C       CMP #$2C        ','
C6B2    F0 14       BEQ $C6C8
C6B4    C9 3D       CMP #$3D        '='
C6B6    F0 10       BEQ $C6C8
C6B8    EE 4B 02    INC $024B       increment length of name
C6BB    E8          INX
C6BC    A9 0F       LDA #$0F        15
C6BE    CD 4B 02    CMP $024B
C6C1    90 05       BCC $C6C8       greater?
C6C3    EC 74 02    CPX $0274       end of input line?
C6C6    90 E5       BCC $C6AD
C6C8    8E 76 02    STX $0276
C6CB    68          PLA
C6CC    AA          TAX             pointer to end of name
C6CD    60          RTS

*******************************
C6CE    A5 83       LDA $83
C6D0    48          PHA             secondary address and channel no.
```

```
C6D1    A5 82       LDA $82
C6D3    48          PHA
C6D4    20 DE C6    JSR $C6DE       create file entry for directory
C6D7    68          PLA
C6D8    85 82       STA $82
C6DA    68          PLA             get data back
C6DB    85 83       STA $83
C6DD    60          RTS

********************************
C6DE    A9 11       LDA #$11        17
C6E0    85 83       STA $83         secondary address
C6E2    20 EB D0    JSR $D0EB       open channel to read
C6E5    20 E8 D4    JSR $D4E8       set buffer pointer
C6E8    AD 53 02    LDA $0253
C6EB    10 0A       BPL $C6F7       not yet last entry?
C6ED    AD 8D 02    LDA $028D
C6F0    D0 0A       BNE $C6FC
C6F2    20 06 C8    JSR $C806       write 'blocks free.'
C6F5    18          CLC
C6F6    60          RTS
C6F7    AD 8D 02    LDA $028D
C6FA    F0 1F       BEQ $C71B
C6FC    CE 8D 02    DEC $028D
C6FF    D0 0D       BNE $C70E
C701    CE 8D 02    DEC $028D
C704    20 8F C3    JSR $C38F       change drive
C707    20 06 C8    JSR $C806       write 'blocks free.'
C70A    38          SEC
C708    4C 8F C3    JMP $C38F       change drive

C70E    A9 00       LDA #$00
C710    8D 73 02    STA $0273       drive no. for header, hi-byte
C713    8D 8D 02    STA $028D
C716    20 B7 C7    JSR $C7B7       write header
C719    38          SEC
C71A    60          RTS

C71B    A2 18       LDX #$18
C71D    A0 1D       LDY #$1D
C71F    B1 94       LDA ($94),Y     number of blocks hi
C721    8D 73 02    STA $0273       in buffer
C724    F0 02       BEQ $C728       zero?
C726    A2 16       LDX #$16
C728    88          DEY
C729    B1 94       LDA ($94),Y     number of blocks lo
C72B    8D 72 02    STA $0272       in buffer
C72E    E0 16       CPX #$16
C730    F0 0A       BEQ $C73C
C732    C9 0A       CMP #$0A        10
C734    90 06       BCC $C73C
C736    CA          DEX
C737    C9 64       CMP #$64        100
C739    90 01       BCC $C73C
C73B    CA          DEX
```

```
C73C    20 AC C7    JSR $C7AC      erase buffer
C73F    B1 94       LDA ($94),Y    file type
C741    48          PHA
C742    0A          ASL A          bit 7 in carry
C743    10 05       BPL $C74A      bit 6 not set?
C745    A9 3C       LDA #$3C       '<' for protected file
C747    9D B2 02    STA $02B2,X    write behind file type
C74A    68          PLA
C74B    29 0F       AND #$0F       isolate bits 0-3
C74D    A8          TAY            as file type marker
C74E    B9 C5 FE    LDA $FEC5,Y    3rd letter of the file type
C751    9D B1 02    STA $02B1,X    in buffer
C754    CA          DEX
C755    B9 C0 FE    LDA $FEC0,Y    2nd letter of file type
C758    9D B1 02    STA $02B1,X    in buffer
C75B    CA          DEX
C75C    B9 BB FE    LDA $FEBB,Y    1st letter of file type
C75F    9D B1 02    STA $02B1,X    in buffer
C762    CA          DEX
C763    CA          DEX
C764    B0 05       BCS $C76B      file not closed?
C766    A9 2A       LDA #$2A       '*'
C768    9D B2 02    STA $02B2,X    before file type in buffer
C76B    A9 A0       LDA #$A0       pad with 'shift blank'
676D    9D B1 02    STA $02B1,X    in buffer
C770    CA          DEX
C771    A0 12       LDY #$12
C773    B1 94       LDA ($94),Y    filenames
C775    9D B1 02    STA $02B1,X    write in buffer
C778    CA          DEX
C779    88          DEY
C77A    C0 03       CPY #$03
C77C    B0 F5       BCS $C773
C77E    A9 22       LDA #$22       '='
C780    9D B1 02    STA $02B1,X    write before file type
C783    E8          INX
C784    E0 20       CPX #$20
C786    B0 0B       BCS $C793
C788    BD B1 02    LDA $02B1,X    character from buffer
C78B    C9 22       CMP #$22       '='?
C78D    F0 04       BEQ $C793
C78F    C9 A0       CMP #$A0       'shift blank' at end of name
C791    D0 F0       BNE $C783
C793    A9 22       LDA #$22       fill through '='
C795    9D B1 02    STA $02B1,X
C798    E8          INX
C799    E0 20       CPX #$20
C89B    B0 0A       BCS $C7A7
C79D    A9 7F       LDA #$7F       bit 7
C79F    3D B1 02    AND $02B1,X
C7A2    9D B1 02    STA $02B1,X    erase in the remaining chars
C7A5    10 F1       BPL $C798
C7A7    20 B5 C4    JSR $C4B5      search for next directory entry
C7AA    38          SEC
C7AB    60          RTS
```

```
********************************  erase directory buffer
C7AC    A0 1B       LDY #$1B
C7AE    A9 20       LDA #$20        ' ' blank
C7B0    99 B0 02    STA $02B0,Y     write in buffer
C7B3    88          DEY
C7B4    D0 FA       BNE $C7B0
C7B6    60          RTS

********************************  create header with disk name
C7B7    20 19 F1    JSR $F119       initialize if needed
C7BA    20 DF F0    JSR $F0DF       read disk name
C7BD    20 AC C7    JSR $C7AC       erase buffer
C7C0    A9 FF       LDA #$FF
C7C2    85 6F       STA $6F
C7C4    A6 7F       LDX $7F         drive number
C7C6    8E 72 02    STX $0272       as block no. lo in buffer
C7C9    A9 00       LDA #$00
C7CB    8D 73 02    STA $0273       block number lo
C7CE    A6 F9       LDX $F9         buffer number
C7D0    BD E0 FE    LDA $FEE0,X     hi-byte of the buffer address
C7D3    85 95       STA $95
C7D5    AD 88 FE    LDA $FE88       $90, position of disk name
C7D8    85 94       STA $94         save
C7DA    A0 16       LDY #$16
C7DC    B1 94       LDA ($94),Y     pad buffer with 'shift blank'
C7DE    C9 A0       CMP #$A0
C7E0    D0 0B       BNE $C7ED
C7E2    A9 31       LDA #$31        '1'
C7E4    2C          .BYTE $2C
C7E5    B1 94       LDA ($94),Y     character from buffer
C7E7    C9 A0       CMP #$A0        compare with 'shift blank'
C7E9    D0 02       BNE $C7ED
C7EB    A9 20       LDA #$20        ' ' blank
C7ED    99 B3 02    STA $02B3       in buffer
C7F0    88          DEY
C7F1    10 F2       BPL $C7E5
C7F3    A9 12       LDA #$12        'RVS ON'
C7F5    8D B1 02    STA $02B1       in buffer
C7F8    A9 22       LDA #$22        '"'
C7FA    8D B2 02    STA $02B2       write before
C7FD    8D C3 02    STA $02C3       and after disk name
C800    A9 20       LDA #$20        ' ' blank
C802    8D C4 02    STA $02C4       behind it
C805    60          RTS

********************************  create last line
C806    20 AC C7    JSR $C7AC       erase buffer
C809    A0 0B       LDY #$0B        12 characters
C80B    B9 17 C8    LDA $C817,Y     'blocks free.'
C80E    99 B1 02    STA $02B1,Y     write in buffer
C811    88          DEY
C812    10 F7       BPL $C80B
C814    4C 4D EF    JMP $EF4D       number of free blocks in front
```

```
********************************
C817    42 4C 4F 43 4B 53 20 46    'blocks f'
C81F    52 45 45 2E                'ree.'

********************************   S command   'scratch'
C823    20 98 C3    JSR $C398      ascertain file type
C826    20 20 C3    JSR $C320      get drive number
C829    20 CA C3    JSR $C3CA      initialize drive if needed
C82C    A9 00       LDA #$00
C82E    85 86       STA $86        counter for erased files
C830    20 9D C4    JSR $C49D      search for file in directory
C833    30 3D       BMI $C872      not found?
C835    20 B7 DD    JSR $DDB7      is file open
C838    90 33       BCC $C86D      yes
C83A    A0 00       LDY #$00
C83C    B1 94       LDA ($94),Y    file type
C83E    29 40       AND #$40       scratch protect
C840    D0 2B       BNE $C86D      yes
C842    20 B6 C8    JSR $C8B6      erase file and note in directory
C845    A0 13       LDY #$13
C847    B1 94       LDA ($94),Y    track no. of the first side-sector
C849    F0 0A       BEQ $C855      none present?
C84B    85 80       STA $80        note track number
C84D    C8          INY
C84E    B1 94       LDA ($94),Y    and sector number
C850    85 81       STA $81
C852    20 7D C8    JSR $C87D      erase side-sector
C855    AE 53 02    LDX $0253      file number
C858    A9 20       LDA #$20
C85A    35 E7       AND $E7,X      bit 5 set?
C85C    D0 0D       BNE $C86B      yes, file not closed
C85E    BD 80 02    LDA $0280,X    get track
C861    85 80       STA $80
C863    BD 85 02    LDA $0285,X    and sector
C866    85 81       STA $81
C868    20 7D C8    JSR $C87D      erase file
C86B    E6 86       INC $86        increment number of erased files
C86D    20 8B C4    JSR $C48B      search for next file
C870    10 C3       BPL $C835      if present, erase
C872    A5 86       LDA $86        number of erased files
C874    85 80       STA $80        save as 'track'
C876    A9 01       LDA #$01       1 as disk status
C878    A0 00       LDY #$00       0 as 'sector'
C87A    4C A3 C1    JMP $C1A3      message 'files scratched'

********************************   erase file
C87D    20 5F EF    JSR $EF5F      free block in BAM
C880    20 75 D4    JSR $D475
C883    20 19 F1    JSR $F119      get buffer number in BAM
C886    B5 A7       LDA $A7,X
C888    C9 FF       CMP #$FF
C88A    F0 08       BEQ $C894
C88C    AD F9 02    LDA $02F9
C88F    09 40       ORA #$40
C891    8D F9 02    STA $02F9
```

```
C894    A9 00       LDA #$00
C896    20 C8 D4    JSR $D4C8       buffer pointer to zero
C899    20 56 D1    JSR $D156       get track
C89C    85 80       STA $80
C89E    20 56 D1    JSR $D156       get sector
C8A1    85 81       STA $81
C8A3    A5 80       LDA $80         track number
C8A5    D0 06       BNE $C8AD       not equal to zero
C8A7    20 F4 EE    JSR $EEF4       write BAM
C8AA    4C 27 D2    JMP $D227       close channel

C8AD    20 5F EF    JSR $EF5F       free block in BAM
C8B0    20 4D D4    JSR $D44D       read next block
C8B3    4C 94 C8    JMP $C894       and continue

******************************      erase directory entry
C8B6    A0 00       LDY #$00
C8B8    98          TYA
C8B9    91 94       STA ($94),Y     set file type to zero
C8BB    20 5E DE    JSR $DE5E       write block
C8BE    4C 99 D5    JMP $D599       and check

******************************      D-command   'backup'
C8C1    A9 31       LDA #$31
C8C3    4C C8 C1    JMP $C1C8       31, 'syntax error'

******************************      format diskette
C8C6    A9 4C       LDA #$4C        JMP-command
C8C8    8D 00 06    STA $0600
C8CB    A9 C7       LDA #$C7
C8CD    8D 01 06    STA $0601       JMP $FAC7 in $600 to $602
C8D0    A9 FA       LDA #$FA
C8D2    8D 02 06    STA $0602
C8D5    A9 03       LDA #$03
C8D7    20 D3 D6    JSR $D6D3       set track and sector number
C8DA    A5 7F       LDA $7F         drive number
C8DC    09 E0       ORA #$E0        command code for formatting
C8DE    85 03       STA $03         transmit
C8E0    A5 03       LDA $03
C8E2    30 FC       BMI $C8E0       wait until formatting done
C8E4    C9 02       CMP #$02
C8E6    90 07       BCC $C8EF       smaller than two, then ok
C8E8    A9 03       LDA #$03
C8EA    A2 00       LDX #$00
C8EC    4C 0A E6    JMP $E60A       21, 'read error'
C8EF    60          RTS

******************************      C-command   'copy'
C8F0    A9 E0       LDA #$E0
C8F2    8D 4F 02    STA $024F
C8F5    20 D1 F0    JSR $F0D1
C8F8    20 19 F1    JSR $F119       get buffer number of BAM
C8FB    A9 FF       LDA #$FF
C8FD    95 A7       STA $A7,X
C8FF    A9 0F       LDA #$0F
```

```
C901    8D 56 02    STA $0256
C904    20 E5 C1    JSR $C1E5      check input line
C907    D0 03       BNE $C90C
C909    4C C1 C8    JMP $C8C1      31, 'syntax error'

C90C    20 F8 C1    JSR $C1F8      check input
C90F    20 20 C3    JSR $C320      test drive number
C912    AD 8B 02    LDA $028B      flag for syntax check
C915    29 55       AND #$55
C917    D0 0F       BNE $C928
C919    AE 7A 02    LDX $027A
C91C    BD 00 02    LDA $0200,X    character of the command
C91F    C9 2A       CMP #$2A       '*'
C921    D0 05       BNE $C928
C923    A9 30       LDA #$30
C925    4C C8 C1    JMP $C1C8      30, 'syntax error'

C928    AD 8B 02    LDA $028B      syntax flag
C92B    29 D9       AND #$D9
C92D    D0 F4       BNE $C923      30, 'syntax error'
C92F    4C 52 C9    JMP $C952
C932    A9 00       LDA #$00
C934    8D 58 02    STA $0258
C937    8D 8C 02    STA $028C      number of drives
C93A    8D 80 02    STA $0280      track number in directory
C93D    8D 81 02    STA $0281
C940    A4 E3       LDA $E3
C942    29 01       AND #$01
C944    85 7F       STA $7F        drive number
C946    09 01       ORA #$01
C948    8D 91 02    STA $0291
C94B    AD 7B 02    LDA $027B
C94E    8D 7A 02    STA $027A
C951    60          RTS

C952    20 4F C4    JSR $C44F      search for file in directory
C955    AD 78 02    LDA $0278      number of filenames in command
C958    C9 03       CMP #$03       smaller than three?
C95A    90 45       BCC $C9A1      yes
C95C    A5 E2       LDA $E2        first drive number
C95E    C5 E3       CMP $E3        second drive number
C960    D0 3F       BNE $C9A1      not on same drive?
C962    A5 DD       LDA $DD        directory block of the 1st file
C964    C5 DE       CMP $DE        same dir block as second file?
C966    D0 39       BNE $C9A1      no
C968    A5 D8       LDA $D8        directory sector of first file
C96A    C5 D9       CMP $D9        same dir sector as second file?
C96C    D0 33       BNE $C9A1      no
C96E    20 CC CA    JSR $CACC      is file present
C971    A9 01       LDA #$01
C973    8D 79 02    STA $0279
C976    20 FA C9    JSR $C9FA
C979    20 25 D1    JSR $D125      get data type
C97C    F0 04       BEQ $C982      rel-file?
C97E    C9 02       CMP #$02       prg-file
```

```
C980    D0 05       BNE $C987     no
C982    A9 64       LDA #$64
C984    20 C8 C1    JSR $C1C8     64, 'file type mismatch'
C987    A9 12       LDA #$12      18
C989    85 83       STA $83       secondary address
C98B    AD 3C 02    LDA $023C
C98E    8D 3D 02    STA $023D
C991    A9 FF       LDA #$FF
C993    8D 3C 02    STA $023C
C996    20 2A DA    JSR $DA2A     prepare append
C999    A2 02       LDX #$02
C99B    20 B9 C9    JSR $C9B9     copy file
C99E    4C 94 C1    JMP $C194     done

C9A1    20 A7 C9    JSR $C9A7     copy file
C9A4    4C 94 C1    JMP $C194     done

C9A7    20 E7 CA    JSR $CAE7
C9AA    A4 E2       LDA $E2       drive no. of first file
C9AC    29 01       AND #$01
C9AE    85 7F       STA $7F       drive number
C9B0    20 86 D4    JSR $D486
C9B3    20 E4 D6    JSR $D6E4     enter file in directory
C9B6    AE 77 02    LDX $0277
C9B9    8E 79 02    STX $0279
C9BC    20 FA C9    JSR $C9FA
C9BF    A9 11       LDA #$11      17
C9C1    85 83       STA $83
C9C3    20 EB D0    JSR $D0EB
C9C6    20 25 D1    JSR $D125     get data type
C9C9    D0 03       BNE $C9CE     no rel-file?
C9CB    20 53 CA    JSR $CA53
C9CE    A9 08       LDA #$08
C9D0    85 F8       STA $F8
C9D2    4C D8 C9    JMP $C9D8

C9D5    20 9B CF    JSR $CF9B     write byte in buffer
C9D8    20 35 CA    JSR $CA35     and get byte
C9DB    A9 80       LDA #$80
C9DD    20 A6 DD    JSR $DDA6     test bit 7
C9E0    F0 F3       BEQ $C9D5     not set?
C9E2    20 25 D1    JSR $D125     check file type
C9E5    F0 03       BEQ $C9EA     rel-file?
C9E7    20 9B CF    JSR $CF9B     get data byte in buffer
C9EA    AE 79 02    LDX $0279
C9ED    E8          INX
C9EE    EC 78 02    CPX $0278
C9F1    90 C6       BCC $C9B9
C9F3    A9 12       LDA #$12      18
C9F5    85 83       STA $83
C9F7    4C 02 DB    JMP $DB02     close channel

C9FA    AE 79 02    LDX $0279
C9FD    B5 E2       LDA $E2,X     drive number
C9FF    29 01       AND #$01
```

```
CA01    85 7F       STA $7F       save
CA03    AD 85 FE    LDA $FE85     18, directory track
CA06    85 80       STA $80       save
CA08    B5 D8       LDA $D8,X     directory sector
CA0A    85 81       STA $81
CA0C    20 75 D4    JSR $D475     read block
CA0F    AE 79 02    LDX $0279
CA12    B5 DD       LDA $DD,X     pointer in block
CA14    20 C8 D4    JSR $D4C8     set buffer pointer
CA17    AE 79 02    LDX $0279
CA1A    B5 E7       LDA $E7,X     file type
CA1C    29 07       AND #$07      isolate
CA1E    8D 4A 02    STA $024A     and save
CA21    A9 00       LDA #$00
CA23    8D 58 02    STA $0258
CA26    20 A0 D9    JSR $D9A0     get parameters for rel-file
CA29    A0 01       LDY #$01
CA2B    20 25 D1    JSR $D125     get file type
CA2E    F0 01       BEQ $CA31     rel-file?
CA30    C8          INY
CA31    98          TYA
CA32    4C C8 D4    JMP $D4C8     set buffer pointer
CA35    A9 11       LDA #$11      17
CA37    85 83       STA $83
CA39    20 9B D3    JSR $D39B     open channel and get byte
CA3C    85 85       STA $85
CA3E    A6 82       LDX $82       channel number
CA40    B5 F2       LDA $F2,X
CA42    29 08       AND #$08      isolate end marker
CA44    85 F8       STA $F8
CA46    D0 0A       BNE $CA52     not set?
CA48    20 25 D1    JSR $D125     get data type
CA4B    F0 05       BEQ $CA52     rel-file?
CA4D    A9 80       LDA #$80
CA4F    20 97 DD    JSR $DD97     set bit 7
CA52    60          RTS

CA53    20 D3 D1    JSR $D1D3     set drive number
CA56    20 CB E1    JSR $E1CB
CA59    A5 D6       LDA $D6
CA5B    48          PHA
CA5C    A5 D5       LDA $D5
CA5E    48          PHA
CA5F    A9 12       LDA #$12      18
CA61    85 83       STA $83
CA63    20 07 D1    JSR $D107     open write channel
CA66    20 D3 D1    JSR $D1D3     set drive number
CA69    20 CB E1    JSR $E1CB
CA6C    20 9C E2    JSR $E29C
CA6F    A5 D6       LDA $D6
CA71    85 87       STA $87
CA73    A5 D5       LDA $D5
CA75    85 86       STA $86
CA77    A9 00       LDA #$00
CA79    85 88       STA $88
```

```
CA7B    85 D4      STA $D4
CA7D    85 D7      STA $D7
CA7F    68         PLA
CA80    85 D5      STA $D5
CA82    68         PLA
CA83    85 D6      STA $D6
CA85    4C 3B E3   JMP $E33B

******************************  R-command,   'rename'
CA88    20 20 C3   JSR $C320    get drive no. from command line
CA8B    A5 E3      LDA $E3
CA8D    29 01      AND #$01
CA8F    85 E3      STA $E3      2nd drive number
CA91    C5 E2      CMP $E2      compare with 1st drive number
CA93    F0 02      BEQ $CA97    same?
CA95    09 80      ORA #$80
CA97    85 E2      STA $E2
CA99    20 4F C4   JSR $C44F    search for file in directory
CA9C    20 E7 CA   JSR $CAE7    does name exist?
CA9F    A5 E3      LDA $E3
CAA1    29 01      AND #$01
CAA3    85 7F      STA $7F      drive number
CAA5    A5 D9      LDA $D9
CAA7    85 81      STA $81      sector number
CAA9    20 57 DE   JSR $DE57    read block from directory
CAAC    20 99 D5   JSR $D599    ok?
CAAF    A5 DE      LDA $DE      pointer to directory entry
CAB1    18         CLC
CAB2    69 03      ADC #$03     pointer plus 3 to file name
CAB4    20 C8 D4   JSR $D4C8    set buffer pointer
CAB7    20 93 DF   JSR $DF93    get buffer number
CABA    A8         TAY
CABB    AE 7A 02   LDX $027A
CABE    A9 10      LDA #$10     16 characters
CAC0    20 6E C6   JSR $C66E    write name in buffer
CAC3    20 5E DE   JSR $DE5E    write block to directory
CAC6    20 99 D5   JSR $D599    ok?
CAC9    4C 94 C1   JMP $C194    done, prepare disk status

******************************  check if file present
CACC    A5 E8      LDA $E8      file type
CACE    29 07      AND #$07
CAD0    8D 4A 02   STA $024A    save
CAD3    AE 78 02   LDX $0278
CAD6    CA         DEX
CAD7    EC 77 02   CPX $0277
CADA    90 0A      BCC $CAE6
CADC    BD 80 02   LDA $0280,X  track number
CADF    D0 F5      BNE $CAD6    not zero?
CAE1    A9 62      LDA #$62
CAE3    4C C8 C1   JMP $C1C8    62, 'file not found'
CAE6    60         RTS

CAE7    20 CC CA   JSR $CACC    does file exist with old name?
CAEA    8D 80 02   LDA $0280,X  track number of new file
```

```
CAED    F0 05       BEQ $CAF4       file erased?
CAEF    A9 63       LDA #$63
CAF1    4C C8 C1    JMP $C1C8       63, 'file exists'
CAF4    CA          DEX
CAF5    10 F3       BPL $CAEA
CAF7    60          RTS

********************************    M-command,   'memory'
CAF8    AD 01 02    LDA $0201       2nd character from buffer
CAFB    C9 2D       CMP #$2D        '-'
CAFD    D0 4C       BNE $CB4B
CAFF    AD 03 02    LDA $0203
CB02    85 6F       STA $6F         address in $6F/$70
CB04    AD 04 02    LDA $0204
CB07    85 70       STA $70
CB09    A0 00       LDY #$00
CB0B    AD 02 02    LDA $0202       3rd character from buffer
CB0E    C9 52       CMP #$52        'R'
CB10    F0 0E       BEQ $CB20       to memory read
CB12    20 58 F2    JSR $F258       (RTS)
CB15    C9 57       CMP #$57        'W'
CB17    F0 37       BEQ $CB50       to memory write
CB19    C9 45       CMP #$45        'E'
CB1B    D0 2E       BNE $CB4B
CB1D    6C 6F 00    JMP ($006F)     memory-execute

********************************    M-R,  'Memory-Read'
CB20    B1 6F       LDA ($6F),Y     read byte
CB22    85 85       STA $85
CB24    AD 74 02    LDA $0274       length of command line
CB27    C9 06       CMP #$06        less than 6?
CB29    90 1A       BCC $CB45       yes
CB2B    AE 05 02    LDX $0205       number
CB2E    CA          DEX
CB2F    F0 14       BEQ $CB45       only one byte?
CB31    8A          TXA             number of bytes
CB32    18          CLC
CB33    65 6F       ADC $6F         plus start address
CB35    E6 6F       INC $6F
CB37    8D 49 02    STA $0249       end pointer
CB3A    A5 6F       LDA $6F
CB3C    85 A5       STA $A5         buffer pointer for error message
CB3E    A5 70       LDA $70         set to start address for 'M-R'
CB40    85 A6       STA $A6
CB42    4C 43 D4    JMP $D443       byte out

CB45    20 EB D0    JSR $D0EB       open read channel
CB48    4C 3A D4    JMP $D43A       byte out

CB4B    A9 31       LDA #$31
CB4D    4C C8 C1    JMP $C1C8       31, 'syntax error'

********************************    M-W,  'memory-write'
CB50    B9 06 02    LDA $0206,Y     read character
CB53    91 6F       STA ($6F),Y     and save
```

```
CB55   C8         INY
CB56   CC 05 02   CPY $0205      number of characters
CB59   90 F5      BCC $CB50      all characters?
CB5B   60         RTS

*******************************  U-command,   'user'
CB5C   AC 01 02   LDY $0201      second char
CB5F   C0 30      CPY #$30       '0'
CB61   D0 09      BNE $CB6C      no
CB63   A9 EA      LDA #$EA
CB65   85 6B      STA $6B        ptr to table of user-addresses
CB67   A9 FF      LDA #$FF       $FFEA
CB69   85 6C      STA $6C
CB6B   60         RTS

CB6C   20 72 CB   JSR $CB72
CB6F   4C 94 C1   JMP $C194      done, prepare error message

CB72   88         DEY
CB73   98         TYA
CB74   29 0F      AND #$0F       number
CB76   0A         ASL A          times 2
CB77   A8         TAY
CB78   B1 6B      LDA ($6B),Y    as pointer in table
CB7A   85 75      STA $75
C87C   C8         INY            address at $75/$76
CB7D   B1 6B      LDA ($6B),Y
CB7F   85 76      STA $76
CB81   6C 75 00   JMP ($0075)    execute function

*******************************  open direct access channel, '#
CB84   AD 8E 02   LDA $028E      last drive number
CB87   85 7F      STA $7F        drive number
CB89   A5 83      LDA $83        channel number
CB8B   48         PHA
CB8C   20 3D C6   JSR $C63D      check drive and initialize
CB8F   68         PLA
CB90   85 83      STA $83
CB92   AE 74 02   LDX $0274      length of filename
CB95   CA         DEX
CB96   D0 0D      BNE $CBA5      greater than one?
CB98   A9 01      LDA #$01
CB9A   20 E2 D1   JSR $D1E2      layout buffer and channel
CB9D   4C F1 CB   JMP $CBF1      set flags, done

CBA0   A9 70      LDA #$70
CBA2   4C C8 C1   JMP $C1C8      70, 'no channel'
CBA5   A0 01      LDY #$01
CBA7   20 7C CC   JSR $CC7C      get buffer number
CBAA   AE 85 02   LDX $0285      buffer number
CBAD   E0 05      CPX #$05       bigger than 5?
CBAF   B0 EF      BCS $CBA0      70, 'no channel'
CBB1   A9 00      LDA #$00
CBB3   85 6F      STA $6F
CBB5   85 70      STA $70
```

```
CBB7    38          SEC
CBB8    26 6F       ROL $6F
CBBA    26 70       ROL $70
CBBC    CA          DEX
CBBD    10 F9       BPL $CBB8
CBBF    A5 6F       LDA $6F
CBC1    2D 4F 02    AND $024F
CBC4    D0 DA       BNE $CBA0
CBC6    A5 70       LDA $70
CBC8    2D 50 02    AND $0250
CBCB    D0 D3       BNE $CBA0
CBCD    A5 6F       LDA $6F
CBCF    0D 4F 02    ORA $024F
CBD2    8D 4F 02    STA $024F
CBD5    A5 70       LDA $70
CBD7    0D 50 02    ORA $0250
CBDA    8D 50 02    STA $0250
CBDD    A9 00       LDA #$00
CBDF    20 E2 D1    JSR $D1E2      search channel
CBE2    A6 82       LDX $82        channel number
CBE4    AD 85 02    LDA $0285      buffer number
CBE7    95 A7       STA $A7,X
CBE9    AA          TAX
CBEA    A5 7F       LDA $7F        drive number
CBEC    95 00       STA $00,X
CBEE    9D 5B 02    STA $025B,X
CBF1    A6 83       LDX $83        secondary address
CBF3    BD 2B 02    LDA $022B,X
CBF6    09 40       ORA #$40       set READ and WRITE flags
CBF8    9D 2B 02    STA $022B,X
CBFB    A4 82       LDY $82        channel number
CBFD    A9 FF       LDA #$FF
CBFF    99 44 02    STA $0244,Y    end pointer
CC02    A9 89       LDA #$89
CC04    99 F2 00    STA $00F2,Y    set READ and WRITE flags
CC07    B9 A7 00    LDA $00A7,Y    buffer number
CC0A    99 3E 02    STA $023E,Y
CC0D    0A          ASL A          times 2
CC0E    AA          TAX
CC0F    A9 01       LDA #$01
CC11    95 99       STA $99,X      buffer pointer to one
CC13    A9 0E       LDA #$0E
CC15    99 EC 00    STA $00EC,Y    flag for direct access
CC18    4C 94 C1    JMP $C194      done

********************************   B-command,  'Block'
CC1B    A0 00       LDY #$00
CC1D    A0 00       LDX #$00
CC1F    A9 2D       LDA #$2D       '-'
CC21    20 68 C2    JSR $C268      search for minus sign
CC24    D0 0A       BNE $CC30      found?
CC26    A9 31       LDA #$31
CC28    4C C8 C1    JMP $C1C8      31, 'syntax error'
```

```
CC2B    A9 30       LDA #$30
CC2D    4C C8 C1    JMP $C1C8      30, 'syntax error'

CC30    8A          TXA
CC31    D0 F8       BNE $CC2B      comma, then error
CC33    A2 05       LDX #$05
CC35    B9 00 02    LDA $0200,Y    char from buffer
CC38    DD 5D CC    CMP $CC5D,X    compare with 'AFRWEP'
CC3B    F0 05       BEQ $CC42      found?
CC3D    CA          DEX
CC3E    10 F8       BPL $CC38      compare with all characters
CC40    30 E4       BMI $CC26      not found, error
CC42    8A          TXA
CC43    09 80       ORA #$80       command number, set bit 7
CC45    8D 2A 02    STA $022A
CC48    20 6F CC    JSR $CC6F      get parameters
CC4B    AD 2A 02    LDA $022A
CC4E    0A          ASL A          number times 2
CC4F    AA          TAX            as index
CC50    BD 64 CC    LDA $CC64,X    address of command hi
CC53    85 70       STA $70
CC55    BD 63 CC    LDA $CC63,X    address lo
CC58    85 6F       STA $6F
CC5A    6C 6F 00    JMP ($006F)    jump to command

********************************   names of the various block cmds
CC5D    41 46 52 57 45 50          'AFRWEP'

********************************   addresses of block commands
CC63    03 CD                      $CD03, B-A
CC65    F5 CC                      $CCF5, B-F
CC67    56 CD                      $CD56, B-R
CC69    73 CD                      $CD73, B-W
CC6B    A3 CD                      $CDA3, B-E
CC6D    BD CD                      $CDBD, B-P

********************************   get parameters for block commands
CC6F    A0 00       LDY #$00
CC71    A2 00       LDX #$00
CC73    A9 3A       LDA #$3A       ':'
CC75    20 68 C2    JSR $C268      test line to colon
CC78    D0 02       BNE $CC7C      found?
CC7A    A0 03       LDY #$03       no, begin at 4th character
CC7C    B9 00 02    LDA $0200,Y    search for separating char
CC7F    C9 20       CMP #$20       ' ' blank
CC81    F0 08       BEQ $CC8B
CC83    C9 1D       CMP #$1D       cursor right
CC85    F0 04       BEQ $CC8B
CC87    C9 2C       CMP #$2C       ',' comma
CC89    D0 07       BNE $CC92
CC8B    C8          INY
CC8C    CC 74 02    CPY $0274      line end?
CC8F    90 EB       BCC $CC7C
CC91    60          RTS
```

```
CC92    20 A1 CC    JSR $CCA1       preserve next parameter
CC95    EE 77 02    INC $0277       increment parameter counter
CC98    AC 79 02    LDY $0279
CC9B    E0 04       CPX #$04        compare with maximum number
CC9D    90 EC       BCC $CC8B
CC9F    B0 8A       BCS $CC2B       30, 'syntax error'
CCA1    A9 00       LDA #$00
CCA3    85 6F       STA $6F
CCA5    85 70       STA $70         erase storage area for decimal #s
CCA7    85 72       STA $72
CCA9    A2 FF       LDX #$FF
CCAB    B9 00 02    LDA $0200,Y     get characters from input buffer
CCAE    C9 40       CMP #$40
CCB0    B0 18       BCS $CCCA       no digits?
CCB2    C9 30       CMP #$30        '0'
CCB4    90 14       BCC $CCCA       no digits?
CCB6    29 0F       AND #$0F        convert ASCII digits to hex
CCB8    48          PHA             and save
CCB9    A5 70       LDA $70
CCBB    85 71       STA $71         move digits one further
CCBD    A4 6F       LDA $6F
CCBF    85 70       STA $70
CCC1    68          PLA
CCC2    85 6F       STA $6F         note read number
CCC4    C8          INY             increment pointer in input buffer
CCC5    CC 74 02    CPY $0274       line end reached
CCC7    90 E1       BCC $CCAB       no
CCCA    8C 79 02    STY $0279       save pointer
CCCD    18          CLC
CCCE    A9 00       LDA #$00
CCD0    E8          INX
CCD1    E0 03       CPX #$03
CCD3    B0 0F       BCS $CCE4       convert hex digits to one byte
CCD5    B4 6F       LDY $6F,X
CCD7    88          DEY
CCD8    30 F6       BMI $CCD0
CCDA    7D F2 CC    ADC $CCF2,X     add decimal value
CCDD    90 F8       BCC $CCD7
CCDF    18          CLC
CCE0    E6 72       INC $72
CCE2    D0 F3       BNE $CCD7
CCE4    48          PHA
CCE5    AE 77 02    LDX $0277       counter for parameters
CCE8    A5 72       LDA $72
CCEA    9D 80 02    STA $0280,X     hi-byte
CCED    68          PLA
CCEE    9D 85 02    STA $0285,X     lo-byte
CCF1    60          RTS

********************************    decimal values
CCF2    01 0A 64                    1, 10, 100

********************************    B-F command,  'Block-Free'
CCF5    20 F5 CD    JSR $CDF5       get track, sector and drive no.
CCF8    20 5F EF    JSR $EF5F       free block
```

```
CCFB   4C 94 C1   JMP $C194     done, prepare error message

********************************
CCFE   A9 01      LDA #$01
CD00   8D F9 02   STA $02F9

******************************** B-A command, 'Block-Allocate'
CD03   20 F5 CD   JSR $CDF5     get track, sector and drive no.
CD06   A5 81      LDA $81       sector
CD08   48         PHA           save
CD09   20 FA F1   JSR $F1FA     find block in BAM
CD0C   F0 0B      BEQ $CD19     block allocated?
CD0E   68         PLA           desired sector
CD0F   C5 81      CMP $81       = next free sector?
CD11   D0 19      BNE $CD2C     no
CD13   20 90 EF   JSR $EF90     allocate block in BAM
CD16   4C 94 C1   JMP $C194     done

CD19   68         PLA
CD1A   A9 00      LDA #$00
CD1C   85 81      STA $81       sector 0
CD1E   E6 80      INC $80       next track
CD20   A5 80      LDA $80       track number
CD22   CD D7 FE   CMP $FED7     36, last track number + 1
CD25   B0 0A      BCS $CD31     >=, then 'no block'
CD27   20 FA F1   JSR $F1FA     find free block in next track
CD2A   F0 EE      BEQ $CD1A     not found, check next track
CD2C   A9 65      LDA #$65
CD2E   20 45 E6   JSR $E645     65, 'no block' next free block
CD31   A9 65      LDA #$65
CD33   20 C8 C1   JSR $C1C8     65,'no block' no more free blocks

********************************
CD36   20 F2 CD   JSR $CDF2     open channel, set parameters
CD39   4C 60 D4   JMP $D460     read block from disk

******************************** get byte from buffer
CD3C   20 2F D1   JSR $D12F     set pointer to buffer
CD3F   A1 99      LDA ($99,X)   get byte
CD41   60         RTS

******************************** read block from disk
CD42   20 36 CD   JSR $CD36     open channel, read block
CD45   A9 00      LDA #$00
CD47   20 C8 D4   JSR $D4C8     set buffer pointer to zero
CD4A   20 3C CD   JSR $CD3C     get a byte from the buffer
CD4D   99 44 02   STA $0244,Y
CD50   A9 89      LDA $89       set read and write flag
CD52   99 F2 00   STA $00F2,Y
CD55   60         RTS

******************************** B-R command, 'Block-Read'
CD56   20 42 CD   JSR $CD42     read block from disk
CD59   20 EC D3   JSR $D3EC     prepare byte from buffer
CD5C   4C 94 C1   JMP $C194     prepare error message
```

```
******************************   U1 command, sub. for 'Block-Read'
CD5F   20 6F CC   JSR $CC6F      get parameters of the command
CD62   20 42 CD   JSR $CD42      read block from disk
CD65   B9 44 02   LDA $0244,Y    end pointer
CD68   99 3E 02   STA $023E,Y    save as data byte
CD6B   A9 FF      LDA #$FF
CD6D   99 44 02   STA $0244,Y    end pointer to $FF
CD70   4C 94 C1   JMP $C194      done, prepare error message

******************************   B-W command,  'Block-Write'
CD73   20 F2 CD   JSR $CDF2      open channel
CD76   20 E8 D4   JSR $D4E8      set buffer pointer
CD79   A8         TAY
CD7A   88         DEY
CD7B   C9 02      CMP #$02       buffer pointer lo less than 2?
CD7D   B0 02      BCS $CD81      no
CD7F   A0 01      LDY #$01
CD81   A9 00      LDA #$00
CD83   20 C8 D4   JSR $D4C8      buffer pointer to zero
CD86   98         TYA
CD87   20 F1 CF   JSR $CFF1      write byte in buffer
CD8A   8A         TXA
CD8B   48         PHA
CD8C   20 64 D4   JSR $D464      write block to disk
CD8F   68         PLA
CD90   AA         TAX
CD91   20 EE D3   JSR $D3EE      get byte from buffer
CD94   4C 94 C1   JMP $C194      done, error message

******************************   U2, sub for 'Block-Write'
CD97   20 6F CC   JSR $CC6F      get command parameters
CD9A   20 F2 CD   JSR $CDF2      open channel
CD9D   20 64 D4   JSR $D464      and write block to disk
CDA0   4C 94 C1   JMP $C194      done

******************************   'B-E' command,  'Block-Execute'
CDA3   20 58 F2   JSR $F258      (RTS)
CDA6   20 36 CD   JSR $CD36      open channel and read block
CDA9   A9 00      LDA #$00
CDAB   85 6F      STA $6F        address low
CDAD   A6 F9      LDX $F9        buffer number
CDAF   BD E0 FE   LDA $FEE0,X    buffer address high
CDB2   85 70      STA $70
CDB4   20 BA CD   JSR $CDBA      execute routine
CDB7   4C 94 C1   JMP $C194      done
CDBA   6C 6F 00   JMP ($006F)    jump to routine

******************************   'B-P' command,  'Block-Pointer'
CDBD   20 D2 CD   JSR $CDD2      open channel, get buffer number
CDC0   A5 F9      LDA $F9        buffer number
CDC2   0A         ASL A          * 2
CDC3   AA         TAX            as index
CDC4   AD 86 02   LDA $0286      pointer value
CDC7   95 99      STA $99,X      save as buffer pointer
```

```
CDC9    20 2F D1    JSR $D12F      prepare a byte in buffer
CDCC    20 EE D3    JSR $D3EE      for output
CDCF    4C 94 C1    JMP $C194      done

********************************   open channel
CDD2    A6 D3       LDX $D3
CDD4    E6 D3       INC $D3
CDD6    BD 85 02    LDA $0285,X    buffer number
CDD9    A8          TAY
CDDA    88          DEY
CDDB    88          DEY
CDDC    C0 0C       CPY #$0C       buffer number smaller than 14?
CDDE    90 05       BCC $CDE5      yes
CDE0    A9 70       LDA #$70
CDE2    4C C8 C1    JMP $C1C8      70, 'no channel'

CDE5    85 83       STA $83        secondary address
CDE7    20 EB D0    JSR $D0EB      open channel
CDEA    B0 F4       BCS $CDE0      already allocated,70 'no channel'
CDEC    20 93 DF    JSR $DF93      buffer number
CDEF    85 F9       STA $F9        set
CDF1    60          RTS

********************************
CDF2    20 D2 CD    JSR $CDD2      check buffer no. and open channel
CDF5    A6 D3       LDX $D3        channel number
CDF7    BD 85 02    LDA $0285,X    buffer address
CDFA    29 01       AND #$01
CDFC    85 7F       STA $7F        drive number
CDFE    BD 87 02    LDA $0287,X
CE01    85 81       STA $81        sector
CE03    BD 86 02    LDA $0286,X
CE06    85 80       STA $80        track
CE08    20 5F D5    JSR $D55F      track and sector ok?
CE0B    4C 00 C1    JMP $C100      turn LED on

********************************   set pointer for rel-file
CE0E    20 2C CE    JSR $CE2C      record number * record length
CE11    20 6E CE    JSR $CE6E      divide by 254
CE14    A5 90       LDA $90        remainder = pointer in data block
CE16    85 D7       STA $D7        data pointer
CE18    20 71 CE    JSR $CE71      divide by 120 = side-sector #
CE1B    E6 D7       INC $D7
CE1D    E6 D7       INC $D7        data ptr + 2 (track/sector ptr)
CE1F    A5 8B       LDA $8B        result of division
CE21    85 D5       STA $D5        equals side-sector number
CE23    A5 90       LDA $90        remainder
CE25    0A          ASL A          times 2
CE26    18          CLC
CE27    69 10       ADC #$10       plus 16
CE29    85 D6       STA $D6        =ptr in side-sector to data block
CE2B    60          RTS

********************************
CE2C    20 D9 CE    JSR $CED9      erase work storage
```

```
CE2F    85 92       STA $92
CE31    A6 82       LDX $82        channel number
CE33    B5 B5       LDA $B5,X      record number lo
CE35    85 90       STA $90
CE37    B5 BB       LDA $BB,X      record number hi
CE39    85 91       STA $91
CE3B    D0 04       BNE $CE41
CE3D    A5 90       LDA $90
CE3F    F0 0B       BEQ $CE4C      record number not zero?
CE41    A5 90       LDA $90
CE43    38          SEC
CE44    E9 01       SBC #$01       then subtract one
CE46    85 90       STA $90
CE48    B0 02       BCS $CE4C
CE4A    C6 91       DEC $91
CE4C    B5 C7       LDA $C7,X      record length
CE4E    85 6F       STA $6F
CE50    46 6F       LSR $6F
CE52    90 03       BCC $CE57
CE54    20 ED CE    JSR $CEED      record number * record length
CE57    20 E5 CE    JSR $CEE5      shift register left
CE5A    A5 6F       LDA $6F
CE5C    D0 F2       BNE $CE50
CE5E    A5 D4       LDA $D4
CE60    18          CLC
CE61    65 8B       ASC $8B
CE63    85 8B       STA $8B
CE65    90 06       BCC $CE6D      result in $8B/$8C/$8D
CE67    E6 8C       INC $8C
CE69    D0 02       BNE $DE6D
CE6B    E6 8D       INC $8D
CE6D    60          RTS

********************************   divide by 254, calculate block #
CE6E    A9 FE       LDA #$FE       254
CE70    2C          .BYTE $2C

********************************   divide by 120, calculate
CE71    A9 78       LDA #$78       side-sector number
CE73    85 6F       STA $6F        divisor
CE75    A2 03       LDX #$03
CE77    B5 8F       LDA $8F,X
CE79    48          PHA
CE7A    B5 8A       LDA $8A,X
CE7C    95 8F       STA $8F,X
CE7E    68          PLA
CE7F    95 8A       STA $8A,X
CE81    CA          DEX
CE82    D0 F3       BNE $CE77
CE84    20 D9 CE    JSR $CED9      erase work storage
CE87    A2 00       LDX #$00
CE89    B5 90       LDA $90,X
CE8B    95 8F       STA $8F,X
CE8D    E8          INX
CE8E    E0 04       CPX #$04
```

```
CE90    90 F7       BCC $CE89
CE92    A9 00       LDA #$00
CE94    85 92       STA $92
CE96    24 6F       BIT $6F
CE98    30 09       BMI $CEA3
CE9A    06 8F       ASL $8F
CE9C    08          PHP
CE9D    46 8F       LSR $8F
CE9F    28          PLP
CEA0    20 E6 CE    JSR $CEE6       shift register 1 left
CEA3    20 ED CE    JSR $CEED       add register 0 to register 1
CEA6    20 E5 CE    JSR $CEE5       shift register 1 left
CEA9    24 6F       BIT $6F
CEAB    30 03       BMI $CEB0
CEAD    20 E2 CE    JSR $CEE2       left-shift register 1 twice
CEB0    A5 8F       LDA $8F
CEB2    18          CLC
CEB3    65 90       ADC $90
CEB5    85 90       STA $90
CEB7    90 06       BCC $CEBF
CEB9    E6 91       INC $91
CEBB    D0 02       BNE $CEBF
CEBD    E6 92       INC $92
CEBF    A5 92       LDA $92
CEC1    05 91       ORA $91
CEC3    D0 C2       BNE $CE87
CEC5    A5 90       LDA $90
CEC7    38          SEC
CEC8    E5 6F       SBC $6F         quotient in $8B/$8C/$8D
CECA    90 0C       BCC $CED8
CECC    E6 8B       INC $8B
CECE    D0 06       BNE $CED6
CED0    E6 8C       INC $8C
CED2    D0 02       BNE $CED6
CED4    85 90       STA $90         remainder in $90
CED8    60          RTS

********************************    erase work storage
CED9    A9 00       LDA #$00
CEDB    85 8B       STA $8B
CEDD    85 8C       STA $8C
CEDF    85 8D       STA $8D
CEE1    60          RTS

********************************    left-shift 3-byte register twice
CEE2    20 E5 CE    JSR $CEE5

********************************    left-shift 3-byte register once
CEE5    18          CLC
CEE6    29 90       ROL $90
CEE8    26 91       ROL $91
CEEA    26 92       ROL $92
CEEC    60          RTS

********************************
```

```
CEED    18          CLC
CEEE    A2 FD       LDX #$FD
CEF0    B5 8E       LDA $8E,X     register $90/$91/$92
CEF2    75 93       ADC $93,X     add to register $8B/$8C/$8D
CEF4    95 8E       STA $8E,X
CEF6    E8          INX
CEF7    D0 F7       BNE $CEF0
CEF9    60          RTS

CEFA    A2 00       LDX #$00
CEFC    8A          TXA
CEFD    95 FA       STA $FA,X
CEFF    E8          INX
CF00    E0 04       CPX #$04
CF02    D0 F8       BNE $CEFC
CF04    A9 06       LDA #$06
CF06    95 FA       STA $FA,X
CF08    60          RTS

CF09    A0 04       LDY #$04
CF0B    A6 82       LDX $82       channel number
CF0D    B9 FA 00    LDA $00FA,Y
CF10    96 FA       STX $FA,Y
CF12    C5 82       CMP $82       channel number
CF14    F0 07       BEQ $CF1D
CF16    88          DEY
CF17    30 E1       BMI $CEFA
CF19    AA          TAX
CF1A    4C 0D CF    JMP $CF0D
CF1D    60          RTS

CF1E    20 09 CF    JSR $CF09
CF21    20 B7 DF    JSR $DFB7
CF24    D0 46       BNE $CF6C
CF26    20 D3 D1    JSR $D1D3     set drive number
CF29    20 8E D2    JSR $D28E
CF2C    30 48       BMI $CF76
CF2E    20 C2 DF    JSR $DFC2
CF31    A5 80       LDA $80       track
CF33    48          PHA
CF34    A5 81       LDA $81       sector
CF36    48          PHA
CF37    A9 01       LDA #$01
CF39    20 F6 D4    JSR $D4F6     get byte 1 from buffer
CF3C    85 81       STA $81       sector
CF3E    A9 00       LDA #$00
CF40    20 F6 D4    JSR $D4F6     get byte 0 from buffer
CF43    85 80       STA $80       track
CF45    F0 1F       BEQ $CF66
CF47    20 25 D1    JSR $D125     check file type
CF4A    F0 0B       BEQ $CF57     rel-file?
CF4C    20 AB DD    JSR $DDAB
CF4F    D0 06       BNE $CF57
CF51    20 8C CF    JSR $CF8C
CF54    4C 5D CF    JMP $CF5D
```

```
CF57    20 8C CF    JSR $CF8C
CF5A    20 57 DE    JSR $DE57
CF5D    68          PLA
CF5E    85 81       STA $81        get sector
CF60    68          PLA
CF61    85 80       STA $80        and track number
CF63    4C 6F CF    JMP $CF6F

CF66    68          PLA
CF67    85 81       STA $81        get back sector
CF69    68          PLA
CF6A    85 80       STA $80        and track number
CF6C    20 8C CF    JSR $CF8C
CF6F    20 93 DF    JSR $DF93
CF72    AA          TAX
CF73    4C 99 D5    JMP $D599      and verify

CF76    A9 70       LDA #$70
CF78    4C C8 C1    JMP $C1C8      70, 'no channel'

CF7B    20 09 CF    JSR $CF09
CF7E    20 B7 DF    JSR $DFB7
CF81    D0 08       BNE $CF8B
CF83    20 8E D2    JSR $D28E
CF86    30 EE       BMI $CF76
CF88    20 C2 DF    JSR $DFC2
CF8B    60          RTS

********************************   change buffer
CF8C    A6 82       LDX $82        channel number
CF8E    B5 A7       LDA $A7,X
CF90    49 80       EOR #$80
CF92    95 A7       STA $A7,X
CF94    B5 AE       LDA $AE,X      rotate bit 7 in table
CF96    49 80       EOR #$80
CF98    95 AE       STA $AE,X
CF9A    60          RTS

********************************   write data byte in buffer
CF9B    A2 12       LDX #$12       channel 18
CF9D    86 83       STX $83
CF9F    20 07 D1    JSR $D107      open write channel
CFA2    20 00 C1    JSR $C100      turn LED on
CFA5    20 25 D1    JSR $D125      check file type
CFA8    90 05       BCC $CFAF      no rel-file
CFAA    A9 20       LDA #$20
CFAC    20 9D DD    JSR $DD9D      change buffer
CFAF    A5 83       LDA $83        secondary address
CFB1    C9 0F       CMP #$0F       15?
CFB3    F0 23       BEQ $CFD8      yes
CFB5    D0 08       BNE $CFBF      no

CFB7    A5 84       LDA $84        secondary address
CFB9    29 8F       AND #$8F
```

```
CFBB    C9 0F       CMP #$0F       greater than 15?
CFBD    B0 19       BCS $CFD8      then input buffer
CFBF    20 25 D1    JSR $D125      check file type
CFC2    B0 05       BCS $CFC9      rel-file or direct access?
CFC4    A5 85       LDA $85        data byte
CFC6    4C 9D D1    JMP $D19D      write in buffer

CFC9    D0 03       BNE $CFCE      direct access file?
CFCB    4C AB E0    JMP $E0AB      write data byte in rel-file

CFCE    A5 85       LDA $85
CFD0    20 F1 CF    JSR $CFF1      write data byte in buffer
CFD3    A4 82       LDY $82        channel number
CFD5    4C EE D3    JMP $D3EE      prepare next byte for output

CFD8    A9 04       LDA #$04       channel 4
CFDA    85 82       STA $82        corresponding input buffer
CFDC    20 E8 D4    JSR $D4E8      set buffer pointer
CFDF    C9 2A       CMP #$2A       40
CFE1    F0 05       BEQ $CFE8      buffer end?
CFE3    A5 85       LDA $85
CFE5    20 F1 CF    JSR $CFF1      write data byte in buffer
CFE8    A5 F8       LDA $F8        end flag set?
CFEA    F0 01       BEQ $CFED      yes
CFEC    60          RTS

CFED    EE 55 02    INC $0255      set command flag
CFF0    60          RTS

******************************     write data byte in buffer
CFF1    48          PHA            save data byte
CFF2    20 93 DF    JSR $DF93      get buffer number
CFF5    10 06       BPL $CFFD      associated buffer?
CFF7    68          PLA
CFF8    A9 61       LDA #$61
CFFA    4C C8 C1    JMP $C1C8      61, 'file not open'
CFFD    0A          ASL A          buffer number times 2
CFFE    AA          TAX            as index
CFFF    68          PLA            data byte
D000    81 99       STA ($99,X)    write in buffer
D002    F6 99       INC $99,X      increment buffer pointer
D004    60          RTS

******************************     I-command, Initialize
D005    20 D1 C1    JSR $C1D1      find drive number
D008    20 42 D0    JSR $D042      load BAM
D00B    4C 94 C1    JMP $C194      prepare disk status

******************************
D00E    20 0F F1    JSR $F10F
D011    A8          TAY
D012    B6 A7       LDX $A7,Y
D014    E0 FF       CPX #$FF
D016    48          PHA
D019    20 8E D2    JSR $D28E
```

```
D01C    AA              TAX
D01D    A9  70          LDA #$70
D021    20  48  E6      JSR $E648       70, 'no channel'
D024    68              PLA
D025    A8              TAY
D026    8A              TXA
D027    09  80          ORA #$80
D029    99  A7  00      STA $00A7,Y
D02C    8A              TXA
D02D    29  0F          AND #$0F
D02F    85  F9          STA $F9
D031    A2  00          LDA #$00
D033    86  81          STX $81         sector 0
D035    AE  85  FE      LDX $FE85       18
D038    86  80          STX $80         track 18
D03A    20  D3  D6      JSR $D6D3       transmit param to disk controller
D03D    A9  B0          LDA #$B0        command code 'read block header'
D03F    4C  8C  D5      JMP $D58C       transmit to disk controller

*******************************         load BAM
D042    20  D1  F0      JSR $F0D1
D045    20  13  D3      JSR $D313
D048    20  0E  D0      JSR $D00E       read block
D04B    A6  7F          LDX $7F         drive number
D04D    A9  00          LDA #$00
D04F    9D  51  02      STA $0251,X     reset flag for 'BAM changed'
D052    8A              TXA
D053    0A              ASL A
D054    AA              TAX
D055    A5  16          LDA $16
D057    95  12          STA $12,X
D059    A4  17          LDA $17         save ID
D05B    95  13          STA $13,X
D05D    20  86  D5      JSR $D586
D060    A5  F9          LDA $F9         buffer number
D062    0A              ASL A
D063    AA              TAX
D064    A9  02          LDA #$02        buffer pointer to $200
D066    95  99          STA $99,X
D068    A1  99          LDA ($99,X)     get character from buffer
D06A    A6  7F          LDX $7F         drive number
D06C    9D  01  01      STA $0101,X
D06F    A9  00          LDA #$00
D071    95  1C          STA $1C,X       flag for write protect
D073    95  FF          STA $FF,X       flag for read error

*******************************         calculate blocks free
D075    20  3A  EF      JSR $EF3A       buffer address to $6D/$6E
D078    A0  04          LDY #$04        begin at position 4
D07A    A9  00          LDA #$00
D07C    AA              TAX
D07D    18              CLC
D07E    71  6D          ADC ($6D),Y     add no. of free blocks per track
D080    90  01          BCC $D083
D082    E8              INX             X as hi-byte
```

```
D083   C8         INY
D084   C8         INY               plus 4
D085   C8         INY
D086   C8         INY
D087   C0 48      CPY #$48          track 18?
D089   F0 F8      BEQ $D083         then skip
D08B   C0 90      CPY #$90          last track number?
D08D   D0 EE      BNE $D07D         no
D08F   48         PHA               lo-byte
D090   8A         TXA               hi-byte
D091   A6 7F      LDX $7F           drive number
D093   9D FC 02   STA $02FC,X       hi-byte to $2FC
D096   68         PLA               lo-byte
D097   9D FA 02   STA $02FA,X       to $2FA
D09A   60         RTS

*******************************
D09B   20 D0 D6   JSR $D6D0         parameters to disk controller
D09E   20 C3 D0   JSR $D0C3         read block
D0A1   20 99 D5   JSR $D599         ok?
D0A4   20 37 D1   JSR $D137         get byte from buffer
D0A7   85 80      STA $80           track
D0A9   20 37 D1   JSR $D137         next byte from buffer
D0AC   85 81      STA $81           sector
D0AE   60         RTS

D0AF   20 9B D0   JSR $D09B
D0B2   A5 80      LDA $80           track
D0B4   D0 01      BNE $D0B7
D0B6   60         RTS
D0B7   20 1E CF   JSR $CF1E         change buffer
D0BA   20 D0 D6   JSR $D6D0         parameters to disk controller
D0BD   20 C3 D0   JSR $D0C3         read block
D0C0   4C 1E CF   JMP $CF1E         change buffer

*******************************     read block
D0C3   A9 80      LDA #$80          code for 'read'
D0C5   D0 02      BNE $D0C9

*******************************     write block
D0C7   A9 90      LDA #$90          code for 'write'
D0C9   8D 4D 02   STA $024D         save
D0CC   20 93 DF   JSR $DF93         get buffer number
D0CF   AA         TAX
D0D0   20 06 D5   JSR $D506         get track/sector, read/write blk
D0D3   8A         TXA
D0D4   48         PHA
D0D5   0A         ASL A             buffer pointer times 2
D0D6   AA         TAX
D0D7   A9 00      LDA #$00
D0D9   95 99      STA $99,X         pointer in buffer to zero
D0DB   20 25 D1   JSR $D125         get file type
D0DE   C9 04      CMP #$04          rel-file or direct access?
D0E0   B0 06      BCS $D0E8         yes
D0E2   F6 B5      INC $B5,X
```

```
D0E4    D0 02       BNE $D0E8      increment block counter
D0E6    F6 BB       INC $BB,X
D0E8    68          PLA
D0E9    AA          TAX
D0EA    60          RTS

********************************   open channel for reading
D0EB    A5 83       LDA $83        secondary address
D0ED    C9 13       CMP #$13       19
D0EF    90 02       BCC $D0F3      smaller?
D0F1    29 0F       AND #$0F
D0F3    C9 0F       CMP #$0F
D0F5    D0 02       BNE $D0F9
D0F7    A9 10       LDA #$10       16
D0F9    AA          TAX
D0FA    38          SEC
D0FB    BD 2B 02    LDA $022B,X
D0FE    30 06       BMI $D106
D100    29 0F       AND #$0F
D102    85 82       STA $82
D104    AA          TAX
D105    18          CLC            flag for ok
D106    60          RTS

********************************   open channel for writing
D107    A4 83       LDA $83        secondary address
D109    C9 13       CMP #$13       19
D10B    90 02       BCC $D10F      smaller?
D10D    29 0F       AND #$0F
D10F    AA          TAX
D110    BD 2B 02    LDA $022B,X    channel number
D113    A8          TAY
D114    0A          ASL A
D115    90 0A       BCC $D121
D117    30 0A       BMI $D123
D119    98          TYA
D11A    29 0F       AND #$0F
D11C    85 82       STA $82
D11E    AA          TAX
D11F    18          CLC            flag for ok
D120    60          RTS

D121    30 F6       BMI $D119
D123    38          SEC            flag for channel allocated
D124    60          RTS

********************************   check for file type 'REL'
D125    A6 82       LDX $82
D127    B5 EC       LDA $EC,X
D129    4A          LSR A
D12A    29 07       AND #$07
D12C    C9 04       CMP #$04       'REL'?
D12E    60          RTS

********************************   get buffer and channel numbers
```

```
D12F    20 93 DF    JSR $DF93      get buffer number
D132    0A          ASL A
D133    AA          TAX
D134    A4 82       LDY $82
D136    60          RTS

******************************     get a byte from buffer
D137    20 2F D1    JSR $D12F      get buffer and channel number
D13A    B9 44 02    LDA $0244,Y    end pointer
D13D    F0 12       BEQ $D151
D13F    A1 99       LDA ($99,X)    get byte from buffer
D141    48          PHA
D142    B5 99       LDA $99,X      buffer pointer
D144    D9 44 02    CMP $0244,Y    equal end pointer?
D147    D0 04       BNE $D14D      no
D149    A9 FF       LDA #$FF
D14B    95 99       STA $99,X      buffer pointer to -1
D14D    68          PLA            data byte
D14E    F6 99       INC $99,X      increment buffer pointer
D150    60          RTS
D151    A1 99       LDA ($99,X)    get character from buffer
D153    F6 99       INC $99,Y      increment buffer pointer
D155    60          RTS

******************************     get byte and read next block
D156    20 37 D1    JSR $D137      get byte from buffer
D159    D0 36       BNE $D191      not last character?
D15B    85 85       STA $85        save data byte
D15D    B9 44 02    LDA $0244,Y    end pointer
D160    F0 08       BEQ $D16A      yes
D162    A9 80       LDA #$80
D164    99 F2 00    STA $00F2,Y    READ-flag
D167    A5 85       LDA $85        data byte
D169    60          RTS

D16A    20 1E CF    JSR $CF1E      change buffer and read next block
D16D    A9 00       LDA #$00
D16F    20 C8 D4    JSR $D4C8      set buffer pointer to zero
D172    20 37 D1    JSR $D137      get first byte from buffer
D175    C9 00       CMP #$00       track number zero
D177    F0 19       BEQ $D192      yes, then last block
D179    85 80       STA $80        save last track number
D17B    20 37 D1    JSR $D137      get next byte
D17E    85 81       STA $81        save as following track
D180    20 1E CF    JSR $CF1E      change buffer and read next block
D183    20 D3 D1    JSR $D1D3      save drive number
D186    20 D0 D6    JSR $D6D0      param to disk controller
D189    20 C3 D0    JSR $D0C3      transmit read command
D18C    20 1E CF    JSR $CF1E      change buffer and read block
D18F    A5 85       LDA $85        get data byte
D191    60          RTS

D192    20 37 D1    JSR $D137      get next byte from buffer
D195    A4 82       LDY $82
D197    99 44 02    STA $0244,Y    save as end pointer
```

```
D19A   A5 85      LDA $85       get data byte back
D19C   60         RTS
******************************  byte in buffer and write block
D19D   20 F1 CF   JSR $CFF1     byte in buffer
D1A0   F0 01      BEQ $D1A3     buffer full?
D1A2   60         RTS

D1A3   20 D3 D1   JSR $D1D3     get drive number
D1A6   20 1E F1   JSR $F11E     find free block in BAM
D1A9   A9 00      LDA #$00
D1AB   20 C8 D4   JSR $D4C8     buffer pointer to zero
D1AE   A5 80      LDA $80
D1B0   20 F1 CF   JSR $CFF1     track number as first byte
D1B3   A5 81      LDA $81
D1B5   20 F1 CF   JSR $CFF1     sector number as second byte
D1B8   20 C7 D0   JSR $D0C7     write block
D1BB   20 1E CF   JSR $CF1E     change buffer
D1BE   20 D0 D6   JSR $D6D0     param to disk controller
D1C1   A9 02      LDA #$02
D1C3   4C C8 D4   JMP $D4C8     buffer pointer to 2

******************************  increment buffer pointer
D1C6   85 6F      STA $6F
D1C8   20 E8 D4   JSR $D4E8     get buffer pointer
D1CB   18         CLC
D1CC   65 6F      ADC $6F
D1CE   95 99      STA $99,X     and increment
D1D0   85 94      STA $94
D1D2   60         RTS

******************************  get drive number
D1D3   20 93 DF   JSR $DF93     get drive number
D1D6   AA         TAX
D1D7   BD 5B 02   LDA $025B,X
D1DA   29 01      AND #$01      isolate drive number
D1DC   85 7F      STA $7F       and save
D1DE   60         RTS

******************************  find write channel and buffer
D1DF   38         SEC           flag for writing
D1E0   B0 01      BCS $D1E3

******************************  find read channel and buffer
D1E2   18         CLC           flag for reading
D1E3   08         PHP           save
D1E4   85 6F      STA $6F       buffer number
D1E6   20 27 D2   JSR $D227     close channel
D1E9   20 7F D3   JSR $D37F     allocate free channel
D1EC   85 82      STA $82       channel number
D1EE   A6 83      LDX $83       secondary address
D1F0   28         PLP
D1F1   90 02      BCC $D1F5     read channel?
D1F3   09 80      ORA #$80      flag for writing
D1F5   9D 2B 02   STA $022B,X   set
D1F8   29 3F      AND #$3F
```

```
D1FA   A8          TAY
D1FB   A9 FF       LDA #$FF      default value
D1FD   99 A7 00    STA $00A7,Y
D200   99 AE 00    STA $00AE,Y   write in associated table
D203   99 CD 00    STA $00CD,Y
D206   C6 6F       DEC $6F       decrement buffer number
D208   30 1C       BMI $D226     done already?
D20A   20 8E D2    JSR $D28E     find buffer
D20D   10 08       BPL $D217     found?
D20F   20 5A D2    JSR $D25A     erase flags in table
D212   A9 70       LDA #$70
D214   4C C8 C1    JMP $C1C8     70, 'no channel'
D217   99 A7 00    STA $00A7,Y   buffer number in table
D21A   C6 6F       DEC $6F       buffer number
D21C   30 08       BMI $D226     already done?
D21E   20 8E D2    JSR $D28E     find buffer
D221   30 EC       BMI $D20F     not found?
D223   99 AE 00    STA $00AE,Y   buffer number in table
D226   60          RTS

******************************   close channel
D227   A5 83       LDA $83       secondary address
D229   C9 0F       CMP #$0F      15?
D22B   D0 01       BNE $D22E     no
D22D   60          RTS           else done already

D22E   A6 83       LDX $83
D230   BD 2B 02    LDA $022B,X   channel number
D233   C9 FF       CMP #$FF      not associated?
D235   F0 22       BEQ $D259     then done
D237   29 3F       AND #$3F
D239   85 82       STA $82       channel number
D23B   A9 FF       LDA #$FF
D23D   9D 2B 02    STA $022B,X   erase association in table
D240   A6 82       LDX $82
D242   A9 00       LDA #$00
D244   95 F2       STA $F2,X     erase READ and WRITE flag
D246   20 5A D2    JSR $D25A     free buffer
D249   A6 82       LDX $82       channel number
D24B   A9 01       LDA #$01      set bit 0
D24D   CA          DEX
D24E   30 03       BMI $D253     shift to correct position
D250   0A          ASL A
D251   D0 FA       BNE $D24D
D253   0D 56 02    ORA $0256     free in allocation register
D256   8D 56 02    STA $0256
D259   60          RTS

******************************   free buffer
D25A   A6 82       LDX $82       channel number
D25C   B5 A7       LDA $A7,X     buffer number
D25E   C9 FF       CMP #$FF
D260   F0 09       BEQ $D26B     not associated?
D262   48          PHA
D263   A9 FF       LDA #$FF
```

```
D265    95 A7       STA $A7,X      erase buffer association
D267    68          PLA
D268    20 F3 D2    JSR $D2F3      erase buffer allocation register
D26B    A6 82       LDX $82        channel number
D26D    B5 AE       LDA $AE,X
D26F    C9 FF       CMP #$FF       associated in second table?
D271    F0 09       BEQ $D27C      no
D273    48          PHA
D274    A9 FF       LDA #$FF
D276    95 AE       STA $AE,X      erase association
D278    68          PLA
D279    20 F3 D2    JSR $D2F3      erase buffer in allocation reg.
D27C    A6 82       LDX $82        channel number
D27E    B5 CD       LDA $CD,X
D280    C9 FF       CMP #$FF       associated in 3rd table?
D282    F0 09       BEQ $D28D      no
D284    48          PHA
D285    A9 FF       LDA #$FF
D287    95 CD       STA $CD,X      erase association
D289    68          PLA
D28A    20 F3 D2    JSR $D2F3      erase buffer in allocation reg
D28D    60          RTS

******************************     find buffer
D28E    98          TYA
D28F    48          PHA
D290    A0 01       LDY #$01
D292    20 BA D2    JSR $D2BA
D295    10 0C       BPL $D2A3
D297    88          DEY
D298    20 BA D2    JSR $D2BA
D29B    10 06       BPL $D2A3
D29D    20 39 D3    JSR $D339
D2A0    AA          TAX
D2A1    30 13       BMI $D2B6
D2A3    B5 00       LDA $00,X
D2A5    30 FC       BMI $D2A3
D2A7    A5 7F       LDA $7F
D2A9    95 00       STA $00,X
D2AB    9D 5B 02    STA $025B,X
D2AE    8A          TXA
D2AF    0A          ASL A
D2B0    A8          TAY
D2B1    A9 02       LDA #$02
D2B3    99 99 00    STA $0099,Y
D2B6    68          PLA
D2B7    A8          TAY
D2B8    8A          TXA
D2B9    60          RTS

D2BA    A2 07       LDX #$07
D2BC    B9 4F 02    LDA $024F,Y
D2BF    3D E9 EF    AND $EFE9,Y    erase bit
D2C2    F0 04       BEQ $D2C8
D2C4    CA          DEX
```

```
D2C5    10 F5       BPL $D2BC
D2C7    60          RTS

D2C8    B9 4F 02    LDA $024F,Y
D2CB    5D E9 EF    EOR $EFE9,X     rotate bit
D2CE    99 4F 02    STA $024F,Y
D2D1    8A          TXA             buffer number
D2D2    88          DEY
D2D3    30 03       BMI $D2D8
D2D5    18          CLC
D2D6    69 08       ADC #$08
D2D8    AA          TAX             buffer number
D2D9    60          RTS
D2DA    A6 82       LDX $82
D2DC    B5 A7       LDA $A7,X
D2DE    30 09       BMI $D2E9
D2E0    8A          TXA
D2E1    18          CLC
D2E2    69 07       ADC #$07
D2E4    AA          TAX
D2E5    B5 A7       LDA $A7,X
D2E7    10 F0       BPL $D2D9
D2E9    C9 FF       CMP #$FF
D2EB    F0 EC       BEQ $D2D9
D2ED    48          PHA
D2EE    A9 FF       LDA #$FF
D2F0    95 A7       STA $A7,X
D2F2    68          PLA
D2F3    29 0F       AND #$0F
D2F5    A8          TAY             buffer number
D2F6    C8          INY
D2F7    A2 10       LDX #$10        16
D2F9    6E 50 02    ROR $0250
D2FC    6E 4F 02    ROR $024F       rotate 16-bit allocation reg.
D2FF    88          DEY
D300    D0 01       BNE $D303
D302    18          CLC             erase bit for buffer
D303    CA          DEX
D304    10 F3       BPL $D2F9
D306    60          RTS

********************************    close all channels
D307    A9 0E       LDA #$0E        14
D309    85 83       STA $83         secondary address
D30B    20 27 D2    JSR $D227       close channel
D30E    C6 83       DEC $83         next secondary address
D310    D0 F9       BNE $D30B
D312    60          RTS

********************************    close channels of other drives
D313    A9 0E       LDA #$0E        14
D315    85 83       STA $83         secondary address
D317    A6 83       LDX $83
D319    BD 2B 02    LDA $022B,X     association table
D31C    C9 FF       CMP #$FF        channel associated?
```

```
D31E    F0 14       BEQ $D334      no
D320    29 3F       AND #$3F
D322    85 82       STA $82        channel number
D324    20 93 DF    JSR $DF93      get buffer number
D327    AA          TAX
D328    BD 5B 02    LDA $025B,X    drive number
D32B    29 01       AND #$01       isolate
D32D    C5 7F       CMP $7F        equal to actual drive number
D32F    D0 03       BNE $D334      no
D331    20 27 D2    JSR $D227      close channel
D334    C6 83       DEC $83        next channel
D336    10 DF       BPL $D317
D338    60          RTS

********************************
D339    A5 6F       LDA $6F
D33B    48          PHA
D33C    A0 00       LDY #$00
D33E    B6 FA       LDX $FA,Y
D340    B5 A7       LDA $A7,X
D342    10 04       BPL $D348
D344    C9 FF       CMP #$FF
D346    D0 16       BNE $D35E
D348    8A          TXA
D349    18          CLC
D34A    69 07       ADC #$07
D34C    AA          TAX
D34D    B5 A7       LDA $A7,X
D34F    10 04       BPL $D355
D351    C9 FF       CMP #$FF
D353    D0 09       BNE $D35E
D355    C8          INY
D356    C0 05       CPY #$05
D358    90 E4       BCC $D33E
D35A    A2 FF       LDX #$FF
D35C    D0 1C       BNE $D37A
D35E    86 6F       STX $6F
D360    29 3F       AND #$3F
D362    AA          TAX
D363    B5 00       LDA $00,X
D365    30 FC       BMI $D363
D367    C9 02       CMP #$02
D369    90 08       BCC $D373
D36B    A6 6F       LDX $6F
D36D    E0 07       CPX #$07
D36F    90 D7       BCC $D348
D371    B0 E2       BCS $D355

D373    A4 6F       LDY $6F
D375    A9 FF       LDA #$FF
D377    99 A7 00    STA $00A7,Y
D37A    68          PLA
D37B    85 6F       STA $6F
D37D    8A          TXA
D37E    60          RTS
```

```
********************************   find channel and allocate
D37F    A0 00      LDY #$00
D381    A9 01      LDA #$01        set bit 0
D383    2C 56 02   BIT $0256
D386    D0 09      BNE $D391       channel free?
D388    C8         INY
D389    0A         ASL A           rotate bit to left
D38A    D0 F7      BNE $D383       all channels checked?
D38C    A9 70      LDA #$70
D38E    4C C8 C1   JMP $C1C8       70, 'no channel'

D391    49 FF      EOR #$FF        rotate bit model
D393    2D 56 02   AND $0256       erase bit
D396    8D 56 02   STA $0256       allocate channel
D399    98         TYA
D39A    60         RTS

********************************   get byte for output
D39B    20 EB D0   JSR $D0EB       open channel for reading
D39E    20 00 C1   JSR $C100       turn LED on
D3A1    20 AA D3   JSR $D3AA       get byte in output register
D3A4    A6 82      LDX $82         channel number
D3A6    BD 3E 02   LDA $023E,X     get byte
D3A9    60         RTS

D3AA    A6 82      LDX $82         channel number
D3AC    20 25 D1   JSR $D125       check file type
D3AF    D0 03      BNE $D3B4       no rel-file?
D3B1    4C 20 E1   JMP $E120       get byte from rel-file

D3B4    A5 83      LDA $83         secondary address
D3B6    C9 0F      CMP #$0F        15
D3B8    F0 5A      BEQ $D414       yes, read error channel
D3BA    B5 F2      LDA $F2,X
D3BC    29 08      AND #$08        end flag set?
D3BE    D0 13      BNE $D3D3       no
D3C0    20 25 D1   JSR $D125       check file type
D3C3    C9 07      CMP #$07        direct access file?
D3C5    D0 07      BNE $D3CE       no
D3C7    A9 89      LDA #$89        set READ and WRITE flag
D3C9    95 F2      STA $F2,X
D3CB    4C DE D3   JMP $D3DE

D3CE    A9 00      LDA #$00
D3D0    95 F2      STA $F2,X       erase READ and WRITE flag
D3D2    60         RTS

D3D3    A5 83      LDA $83         secondary address
D3D5    F0 32      BEQ $D409       zero, LOAD?
D3D7    20 25 D1   JSR $D125       check file type
D3DA    C9 04      CMP #$04        rel-file or direct access?
D3DC    90 22      BCC $D400       no
D3DE    20 2F D1   JSR $D12F       get buffer and channel number
D3E1    B5 99      LDA $99,X       buffer pointer
```

```
D3E3    D9 44 02    CMP $0244,Y    equal end pointer?
D3E6    D0 04       BNE $D3EC      no
D3E8    A9 00       LDA #$00
D3EA    95 99       STA $99,X      buffer pointer to zero
D3EC    F6 99       INC $99,X      increment buffer pointer
D3EE    A1 99       LDA ($99,X)    get byte from buffer
D3F0    99 3E 02    STA $023E,Y    into output register
D3F3    B5 99       LDA $99,X      buffer pointer
D3F5    D9 44 02    CMP $0244,Y    equal end pointer?
D3F8    D0 05       BNE $D3FF      no
D3FA    A9 81       LDA #$81
D3FC    99 F2 00    STA $00F2,Y    set flags
D3FF    60          RTS

D400    20 56 D1    JSR $D156      get byte from buffer
D403    A6 82       LDX $82        channel number
D405    9D 3E 02    STA $023E,X    byte in output register
D408    60          RTS

D409    AD 54 02    LDA $0254      flag for directory?
D40C    F0 F2       BEQ $D400      no
D40E    20 67 ED    JSR $ED67      create directory line
D411    4C 03 D4    JMP $D403

D414    20 E8 D4    JSR $D4E8      set buffer pointer
D417    C9 D4       CMP #$D4
D419    D0 18       BNE $D433
D41B    A5 95       LDA $95
D41D    C9 02       CMP #$02
D41F    D0 12       BNE $D433
D421    A9 0D       LDA #$0D       CR
D423    85 85       STA $85        in output register
D425    20 23 C1    JSR $C123      erase error flags
D428    A9 00       LDA #$00
D42A    20 C1 E6    JSR $E6C1      create 'ok' message
D42D    C6 A5       DEC $A5        set buffer pointer back
D42F    A9 80       LDA #$80       set READ flag
D431    D0 12       BNE $D445

D433    20 37 D1    JSR $D137      get byte from buffer
D436    85 85       STA $85        into output register
D438    D0 09       BNE $D443
D43A    A9 D4       LDA #$D4
D43C    20 C8 D4    JSR $D4C8      set buf ptr in front of error ptr
D43F    A9 02       LDA #$02
D441    95 9A       STA $9A,X      hi-address
D443    A9 88       LDA #$88       set READ flag
D445    85 F7       STA $F7
D447    A5 85       LDA $85        data byte
D449    8D 43 02    STA $0243      into output register
D44C    60          RTS

********************************   read next block
D44D    20 93 DF    JSR $DF93      get buffer number
D450    0A          ASL A          times 2
```

```
D451   AA         TAX
D452   A9 00      LDA #$00
D454   95 99      STA $99,X      buffer pointer to zero
D456   A1 99      LDA ($99,X)    get first byte from buffer
D458   F0 05      BEQ $D45F      no block following?
D45A   D6 99      DEC $99,X      buffer pointer to -1
D45C   4C 56 D1   JMP $D156      read next block
D45F   60         RTS

********************************    read block
D460   A9 80      LDA #$80       command code for reading
D462   D0 02      BNE $D466

********************************    write block
D464   A9 90      LDA #$90       command code for writing
D466   05 7F      ORA $7F        drive number
D468   8D 4D 02   STA $024D      save code
D46B   A5 F9      LDA $F9
D46D   20 D3 D6   JSR $D6D3      param to disk controller
D470   A6 F9      LDX $F9
D472   4C 93 D5   JMP $D593      execute command

********************************    allocate buffer and read block
D475   A9 01      LDA #$01
D477   8D 4A 02   STA $024A      file type to sequential
D47A   A9 11      LDA #$11       17
D47C   85 83      STA $83        secondary address
D47E   20 46 DC   JSR $DC46      allocate buffer and read block
D481   A9 02      LDA #$02
D483   4C C8 D4   JMP $D4C8      buffer pointer to 2

********************************    allocate new block
D486   A9 12      LDA #$12       18
D488   85 83      STA $83        secondary address
D48A   4C DA DC   JMP $DCDA      allocate new block

********************************    write directory block
D48D   20 3B DE   JSR $DE3B      get track and sector number
D490   A9 01      LDA #$01
D492   85 6F      STA $6F        a block
D494   A5 69      LDA $69        save step width 10 for block
D496   48         PHA            allocation
D497   A9 03      LDA #$03
D499   85 69      STA $69
D49B   20 2D F1   JSR $F12D      find free block in BAM
D49E   68         PLA
D49F   85 69      STA $69        get step width back
D4A1   A9 00      LDA #$00
D4A3   20 C8 D4   JSR $D4C8      buffer pointer to zero
D4A6   A5 80      LDA $80
D4A8   20 F1 CF   JSR $CFF1      track number in buffer
D4AB   A5 81      LDA $81
D4AD   20 F1 CF   JSR $CFF1      sector number in buffer
D4B0   20 C7 D0   JSR $D0C7      write block to disk
D4B3   20 99 D5   JSR $D599      and verify
```

```
D4B6    A9 00       LDA #$00
D4B8    20 C8 D4    JSR $D4C8     buffer pointer to zero
D4BB    20 F1 CF    JSR $CFF1     fill buffer with zeroes
D4BE    D0 FB       BNE $D4BB
D4C0    20 F1 CF    JSR $CFF1     zero as following track
D4C3    A9 FF       LDA #$FF
D4C5    4C F1 CF    JMP $CFF1     $FF as number of bytes

********************************  set buffer pointer
D4C8    85 6F       STA $6F       save pointer
D4CA    20 93 DF    JSR $DF93     get buffer number
D4CD    0A          ASL A         times 2
D4CE    AA          TAX
D4CF    B5 9A       LDA $9A,X     buffer pointer hi
D4D1    85 95       STA $95
D4D3    A5 6F       LDA $6F
D4D5    95 99       STA $99,X     buffer pointer lo, new value
D4D7    85 94       STA $94
D4D9    60          RTS

********************************  close internal channel
D4DA    A9 11       LDA #$11      17
D4DC    85 83       STA $83
D4DE    20 27 D2    JSR $D227     close channel
D4E1    A9 12       LDA #$12      18
D4E3    85 83       STA $83
D4E5    4C 27 D2    JMP $D227     close channel

********************************  set buffer pointer
D4E8    20 93 DF    JSR $DF93     get buffer number
D4EB    0A          ASL A
D4EC    AA          TAX
D4ED    B5 9A       LDA $9A,X     buffer pointer hi
D4EF    85 95       STA $95
D4F1    B5 99       LDA $99,X     buffer pointer lo
D4F3    85 94       STA $94
D4F5    60          RTS

********************************  get byte from buffer
D4F6    85 71       STA $71       pointer lo
D4F8    20 93 DF    JSR $DF93     get buffer number
D4FB    AA          TAX
D4FC    BD E0 FE    LDA $FEE0,X   hi-byte buffer address
D4FF    85 72       STA $72       pointer hi
D501    A0 00       LDY #$00
D503    B1 71       LDA ($71),Y   get byte from buffer
D505    60          RTS

********************************  check track and sector numbers
D506    BD 5B 02    LDA $025B,X   command code for disk controller
D509    29 01       AND #$01      drive number
D50B    0D 4D 02    ORA $024D     plus command code
D50E    A8          PHA           save
D50F    86 F9       STX $F9       buffer number
D511    8A          TXA
```

```
D512    0A          ASL A       times 2
D513    AA          TAX
D514    B5 07       LDA $07,X   sector
D516    8D 4D 02    STA $024D   save
D519    B5 06       LDA $06,X   track
D51B    F0 2D       BEQ $D54A   66, 'illegal track or sector'
D51D    CD D7 FE    CMP $FED7   36, highest track number + 1
D520    B0 28       BCS $D54A   66, 'illegal track or sector'
D522    AA          TAX
D523    68          PLA         command code
D524    48          PHA
D525    29 F0       AND #$F0
D527    C9 90       CMP #$90    code for writing?
D529    D0 4F       BNE $D57A   no
D52B    68          PLA
D52C    48          PHA
D52D    4A          LSR A
D52E    B0 05       BCS $D535
D530    AD 01 01    LDA $0101
D533    90 03       BCC $D538
D535    AD 02 01    LDA $0102
D538    F0 05       BEQ $D53F
D53A    CD D5 FE    CMP $FED5   'A', format marker
D53D    D0 33       BNE $D572   73, 'cbm dos v2.6 1541'
D53F    8A          TXA         track number
D540    20 4B F2    JSR $F24B   get maximum sector number
D543    CD 4D 02    CMP $024D   compare with sector number
D546    F0 02       BEQ $D54A   equal, then error
D548    B0 30       BCS $D57A   smaller?
D54A    20 52 D5    JSR $D552   get track and sector number
D54D    A9 66       LDA #$66
D54F    4C 45 E6    JMP $E645   66, 'illegal track or sector'

******************************  get track and sector number
D552    A5 F9       LDA $F9     buffer number
D554    0A          ASL A       *2
D555    AA          TAX         as index
D556    B5 06       LDA $06,X
D558    85 80       STA $80     track
D55A    B5 07       LDA $07,X
D55C    85 81       STA $81     sector
D55E    60          RTS

D55F    A5 80       LDA $80     track
D561    F0 EA       BEQ $D54D   zero, then error
D563    CD D7 FE    CMP $FED7   36, maximum track number + 1
D566    B0 E5       BCS $D54D   66, 'illegal track or sector'
D568    20 4B F2    JSR $F24B   get maximum sector number
D56B    C5 81       CMP $81     sector
D56D    F0 DE       BEQ $D54D
D56F    90 DC       BCC $D54D   error
D571    60          RTS

D572    20 52 D5    JSR $D552   get track and sector numbe
D575    A9 73       LDA #$73
```

```
D577    4C 45 E6    JMP $E645      73, 'cbm dos v2.6 1541'

D57A    A6 F9       LDX $F9        buffer number
D57C    68          PLA
D57D    8D 4D 02    STA $024D      command code for disk controller
D580    95 00       STA $00,X      in command register
D582    9D 5B 02    STA $025B,X    and write in table
D585    60          RTS

********************************   read block
D586    A9 80       LDA #$80       code for read
D588    D0 02       BNE $D58C

********************************   write block
D58A    A9 90       LDA #$90       code for write
D58C    05 7F       ORA $7F        drive number
D58E    A6 F9       LDX $F9        buffer number
D590    8D 4D 02    STA $024D
D593    AD 4D 02    LDA $024D      command code
D596    20 0E D5    JSR $D50E      check track and sector

********************************   verify execution
D599    20 A6 D5    JSR $D5A6      verify execution
D59C    B0 FB       BCS $D599      wait for end
D59E    48          PHA
D59F    A9 00       LDA #$00
D5A1    8D 98 02    STA $0298      erase error flag
D5A4    68          PLA
D5A5    60          RTS

D5A6    B5 00       LDA $00,X      cmd code (bit 7) still in reg?
D5A8    30 1A       BMI $D5C4      yes
D5AA    C9 02       CMP #$02
D5AC    90 14       BCC $D5C2      error-free execution
D5AE    C9 08       CMP #$08       8
D5B0    F0 08       BEQ $D5BA      write protect
D5B2    C9 0B       CMP #$0B       11
D5B4    F0 04       BEQ $D5BA      ID mismatch
D5B6    C9 0F       CMP #$0F       15
D5B8    D0 0C       BNE $D5C6
D5BA    2C 98 02    BIT $0298
D5BD    30 03       BMI $D5C2
D5BF    4C 3F D6    JMP $D63F      create error message
D5C2    18          CLC            execution ended
D5C3    60          RTS

D5C4    38          SEC            execution not yet ended
D5C5    60          RTS

D5C6    98          TYA
D5C7    48          PHA
D5C8    A5 7F       LDA $7F        drive number
D5CA    48          PHA
D5CB    BD 5B 02    LDA $025B,X
```

```
D5CE   29 01      AND #$01      drive number
D5D0   85 7F      STA $7F
D5D2   A8         TAY
D5D3   B9 CA FE   LDA $FECA,Y   bit model for drive
D5D6   8D 6D 02   STA $026D
D5D9   20 A6 D6   JSR $D6A6     read attempt
D5DC   C9 02      CMP #$02
D5DE   B0 03      BCS $D5E3     not ok?
D5E0   4C 6D D6   JMP $D66D     done
D5E3   BD 5B 02   LDA $025B,X   command code
D5E6   29 F0      AND #$F0      isolate
D5E8   48         PHA
D5E9   C9 90      CMP #$90      code for write
D5EB   D0 07      BNE $D5F4     no
D5ED   A5 7F      LDA $7F       drive number
D5EF   09 B8      ORA #$B8
D5F1   9D 5B 02   STA $025B,X
D5F4   24 6A      BIT $6A
D5F6   70 39      BVS $D631
D5F8   A9 00      LDA #$00
D5FA   8D 99 02   STA $0299     cntr for searches next to track
D5FD   8D 9A 02   STA $029A
D600   AC 99 02   LDY $0299     counter
D603   AD 9A 02   LDA $029A
D606   38         SEC
D607   F9 DB FE   SBC $FEDB,Y   constants for read attempts
D60A   8D 9A 02   STA $029A
D60D   B9 DB FE   LDA $FEDB,Y
D610   20 76 D6   JSR $D676     position head next to track
D613   EE 99 02   INC $0299     increment counter
D616   20 A6 D6   JSR $D6A6     read atempt
D619   C9 02      CMP #$02      return message
D61B   90 08      BCC $D625     smaller than 2, ok?
D61D   AC 99 02   LDY $0299     load counter
D620   B9 DB FE   LDA $FEDB,Y   get constants
D623   D0 DB      BNE $D600     not yet zero (table end)?
D625   AD 9A 02   LDA $029A
D628   20 76 D6   JSR $D676     position head
D62B   B5 00      LDA $00,X
D62D   C9 02      CMP #$02      return message
D62F   90 2B      BCC $D65C     ok?
D631   24 6A      BIT $6A
D633   10 0F      BPL $D644
D635   68         PLA           command code
D636   C9 90      CMP #$90      for writing?
D638   D0 05      BNE $D63F     no
D63A   05 7F      ORA $7F       drive number
D63C   9D 5B 02   STA $025B,X   command code in table
D63F   B5 00      LDA $00,X     return message
D641   20 0A E6   JSR $E60A     set error message
D644   68         PLA
D645   2C 98 02   BIT $0298
D648   30 23      BMI $D66D
D64A   48         PHA
D64B   A9 C0      LDA #$C0      command code for head positioning
```

```
D64D    05 7F       ORA $7F       drive number
D64F    95 00       STA $00,X     in command register
D651    B5 00       LDA $00,X
D653    30 FC       BMI $D651     wait for execution
D655    20 A6 D6    JSR $D6A6     attempt command execution again
D658    C9 02       CMP #$02      return message
D65A    B0 D9       BCS $D635     incorrect?
D65C    68          PLA
D65D    C9 90       CMP #$90      command code for writing
D65F    D0 0C       BNE $D66D     no
D661    05 7F       ORA $7F       drive number
D663    9D 5B 02    STA $025B,X   in table
D666    20 A6 D6    JSR $D6A6     attempt execution again
D669    C9 02       CMP #$02      return message
D66B    B0 D2       BCS $D63F     error?
D66D    68          PLA
D66E    85 7F       STA $7F       get drive number back
D670    68          PLA
D671    A8          TAY
D672    B5 00       LDA $00,X     error code
D674    18          CLC           end-of-execution flag
D675    60          RTS

D676    C9 00       CMP #$00
D678    F0 18       BEQ $D692
D67A    30 0C       BMI $D688
D67C    A0 01       LDY #$01
D67E    20 93 D6    JSR $D693     transmit data for head position
D681    38          SEC
D682    E9 01       SBC #$01
D684    D0 F6       BNE $D67C
D686    F0 0A       BEQ $D692

D688    A0 FF       LDY #$FF
D68A    20 93 D6    JSR $D693     transmit data for head position
D68D    18          CLC
D68E    69 01       ADC #$01
D690    D0 F6       BNE $D688
D692    60          RTS

D693    48          PHA
D694    98          TYA
D695    A4 7F       LDY $7F       drive number
D697    99 FE 02    STA $02FE,Y
D69A    D9 FE 02    CMP $02FE,Y   wait for return message from
D69D    F0 FB       BEQ $D69A
D69F    A9 00       LDA #$00      disk controller
D6A1    99 FE 02    STA $02FE,Y
D6A4    68          PLA
D6A5    60          RTS

D6A6    A5 6A       LDA $6A       maximum number of repetitions
D6A8    29 3F       AND #$3F
D6AA    A8          TAY
D6AB    AD 6D 02    LDA $026D     bit for LED
```

```
D6AE    4D 00 1C    EOR $1C00
D6B1    8D 00 1C    STA $1C00
D6B4    BD 5B 02    LDA $025B,X    command
D6B7    95 00       STA $00,X      transmit to disk controller
D6B9    B5 00       LDA $00,X      and return message
D6BB    30 FC       BMI $D6B9      wait
D6BD    C9 02       CMP #$02       ok?
D6BF    90 03       BCC $D6C4      yes
D6C1    88          DEY            decrement counter
D6C2    D0 E7       BNE $D6AB      attempt again
D6C4    48          PHA
D6C5    AD 6D 02    LDA $026D
D6C8    0D 00 1C    ORA $1C00      LED off
D6CB    8D 00 1C    STA $1C00
D6CE    68          PLA
D6CF    60          RTS

******************************     transmit param to disk controller
D6D0    20 93 DF    JSR $DF93      get buffer number
D6D3    0A          ASL A
D6D4    A8          TAY
D6D5    A5 80       LDA $80        track number
D6D7    99 06 00    STA $0006,Y    transmit
D6DA    A5 81       LDA $81        sector number
D6DC    99 07 00    STA $0007,Y    transmit
D6DF    A5 7F       LDA $7F        drive number
D6E1    0A          ASL            times 2
D6E2    AA          TAX
D6E3    60          RTS

******************************     enter file in directory
D6E4    A5 83       LDA $83        secondary address
D6E6    48          PHA
D6E7    A5 82       LDA $82        channel number
D6E9    48          PHA
D6EA    A5 81       LDA $81        sector number
D6EC    48          PHA
D6ED    A5 80       LDA $80        track number
D6EF    48          PHA            save
D6F0    A9 11       LDA #$11
D6F2    85 83       STA $83        secondary address 17
D6F4    20 3B DE    JSR $DE3B      get track and sector number
D6F7    AD 4A 02    LDA $024A      file type
D6FA    48          PHA            save
D6FB    A4 E2       LDA $E2        drive number
D6FD    29 01       AND #$01
D6FF    85 7F       STA $7F        set
D701    A6 F9       LDX $F9        buffer number
D703    5D 5B 02    EOR $025B,X
D706    4A          LSR A
D707    90 0C       BCC $D715      equal drive number?
D709    A2 01       LDX #$01
D70B    8E 92 02    STX $0292      pointer in directory
D70E    20 AC C5    JSR $C5AC      load dir and find first entry
D711    F0 1D       BEQ $D730      not found?
```

```
D713    D0 28       BNE $D73D      found?

D715    AD 91 02    LDA $0291      sector number in directory
D718    F0 0C       BEQ $D726      equal zero
D71A    C5 81       CMP $81        equal sector number?
D71C    F0 1F       BEQ $D73D      yes
D71E    85 81       STA $81        save sector number
D720    20 60 D4    JSR $D460      read block
D723    4C 3D D7    JMP $D73D

D726    A9 01       LDA #$01
D728    8D 92 02    STA $0292      pointer to one
D72B    20 17 C6    JSR $C617      find next entry in directory
D72E    D0 0D       BNE $D73D      found?
D730    20 8D D4    JSR $D48D      write directory block
D733    A5 81       LDA $81        sector number
D735    8D 91 02    STA $0291
D738    A9 02       LDA #$02
D73A    8D 92 02    STA $0292      pointer to 2
D73D    AD 92 02    LDA $0292
D740    20 C8 D4    JSR $D4C8      set buffer pointer
D743    68          PLA
D744    8D 4A 02    STA $024A      file type
D747    C9 04       CMP #$04       rel-file?
D749    D0 02       BNE $D74D      no
D74B    09 80       ORA #$80       set bit 7
D74D    20 F1 CF    JSR $CFF1      and write in buffer
D750    68          PLA
D751    8D 80 02    STA $0280      following track
D754    20 F1 CF    JSR $CFF1      in buffer
D757    68          PLA
D758    8D 85 02    STA $0285      following sector
D75B    20 F1 CF    JSR $CFF1      in buffer
D75E    20 93 DF    JSR $DF93      get buffer number
D761    A8          TAY
D762    AD 7A 02    LDA $027A      pointer to drive number
D765    AA          TAX
D766    A9 10       LDA #$10       16, length of filename
D768    20 6E C6    JSR $C66E      write filename in buffer
D76B    A0 10       LDY #$10
D76D    A9 00       LDA #$00
D76F    91 94       STA ($94),Y    fill with zeroes at pos 16
D771    C8          INY
D772    C0 1B       CPY #$1B       position 27 already?
D774    90 F9       BCC $D76F      no
D776    AD 4A 02    LDA $024A      file type
D779    C9 04       CMP #$04       rel-file
D77B    D0 13       BNE $D790      no
D77D    A0 10       LDY #$10
D77F    AD 59 02    LDA $0259      track
D782    91 94       STA ($94),Y
D784    C8          INY
D785    AD 5A 02    LDA $025A      and sector
D788    91 94       STA ($94),Y    the side-sectors in dir entry
D78A    C8          INY
```

```
D78B    AD 58 02   LDA $0258     record length
D78E    91 94      STA ($94),Y   in directory
D790    20 64 D4   JSR $D464     write block
D793    68         PLA
D794    85 82      STA $82       channel number
D796    AA         TAX
D797    68         PLA
D798    85 83      STA $83       secondary address
D79A    AD 91 02   LDA $0291
D79D    85 D8      STA $D8
D79F    9D 60 02   STA $0260,X
D7A2    AD 92 02   LDA $0292
D7A5    85 DD      STA $DD
D7A7    9D 66 02   STA $0266,X
D7AA    AD 4A 02   LDA $024A     file type
D7AD    85 E7      STA $E7
D7AF    A5 7F      LDA $7F       drive number
D7B1    85 E2      STA $E2
D7B3    60         RTS

******************************** OPEN command, secondary adr <> 15
D7B4    A5 83      LDA $83       secondary address
D7B6    8D 4C 02   STA $024C
D7B9    20 B3 C2   JSR $C283     get line length, erase flags
D7BC    8E 2A 02   STX $022A
D7BF    AE 00 02   LDX $0200     first character from buffer
D7C2    AD 4C 02   LDA $024C     secondary address
D7C5    D0 2C      BNE $D7F3     not equal 0 (LOAD)?
D7C7    E0 2A      CPX #$2A      '*'
D7C9    D0 28      BNE $D7F3
D7CB    A5 7E      LDA $7E       last track number
D7CD    F0 4D      BEQ $D81C
D7CF    85 80      STA $80       track number
D7D1    AD 6E 02   LDA $026E     last drive number
D7D4    85 7F      STA $7F       drive number
D7D6    85 E2      STA $E2
D7D8    A9 02      LDA #$02
D7DA    85 E7      STA $E7       set data type to program
D7DC    AD 6F 02   LDA $026F     last sector number
D7DF    85 81      STA $81       sector
D7E1    20 00 C1   JSR $C100     turn LED on
D7E4    20 46 DC   JSR $DC46     allocate buffer, read block
D7E7    A9 04      LDA #$04      file type
D7E9    05 7F      ORA $7F       drive number
D7EB    A6 82      LDX $82       channel number
D7ED    99 EC 00   STA $00EC,Y   set flag
D7F0    4C 94 C1   JMP $C194     done

D7F3    E0 24      CPX #$24      '$'
D7F5    D0 1E      BNE $D815     no
D7F7    AD 4C 02   LDA $024C     secondary address
D7FA    D0 03      BNE $D7FF     not equal to zero?
D7FC    4C 55 DA   JMP $DA55     OPEN $

D7FF    20 D1 C1   JSR $C1D1     analyze line to end
```

```
D802    AD 85 FE    LDA $FE85    18, directory track
D805    85 80       STA $80      track
D807    A9 00       LDA #$00
D809    85 81       STA $81      sector 0
D80B    20 46 DC    JSR $DC46    allocate buffer, read block
D80E    A5 7F       LDA $7F      drive number
D810    09 02       ORA #$02
D812    4C EB D7    JMP $D7EB    continue as above

D815    E0 23       CPX #$23     '#'
D817    D0 12       BNE $D82B
D819    4C 84 CB    JMP $CB84    open direct access file

D81C    A9 02       LDA #$02
D81E    8D 96 02    STA $0296    file type program
D821    A9 00       LDA #$00
D823    85 7F       STA $7F      drive 0
D825    8D 8E 02    STA $028E
D828    20 42 D0    JSR $D042    load BAM
D82B    20 E5 C1    JSR $C1E5    analyze line
D82E    D0 04       BNE $D834    colon found?
D830    A2 00       LDX #$00
D832    F0 0C       BEQ $D840
D834    8A          TXA          comma found?
D835    F0 05       BEQ $D83C    no
D837    A9 30       LDA #$30
D839    4C C8 C1    JMP $C1C8    30, 'syntax error'

D83C    88          DEY
D83D    F0 01       BEQ $D840
D83F    88          DEY
D840    8C 7A 02    STY $027A    pointer to drive number
D843    A9 8D       LDA #$8D     shift CR
D845    20 68 C2    JSR $C268    analyze line to end
D848    E8          INX
D849    8E 78 02    STX $0278    comma counter
D84C    20 12 C3    JSR $C312    get drive number
D84F    20 CA C3    JSR $C3CA    check drive number
D852    20 9D C4    JSR $C49D    find file entry in directory
D855    A2 00       LDX #$00     default values
D857    8E 58 02    STX $0258    record length
D85A    8E 97 02    STX $0297
D85D    8E 4A 02    STX $024A    file type
D860    E8          INX
D861    EC 77 02    CPX $0277    comma before equal sign?
D864    B0 10       BCS $D876    no
D866    20 09 DA    JSR $DA09    get file type and control mode
D869    E8          INX
D86A    EC 77 02    CPX $0277    additional comma?
D86D    B0 07       BCS $D876    no
D86F    C0 04       CPY #$04
D871    F0 3E       BEQ $D8B1
D873    20 09 DA    JSR $DA09    get file type and control method
D876    AE 4C 02    LDX $024C
D879    86 83       STX $83      secondary address
```

```
D87B    E0 02       CPX #$02       greater than 2?
D87D    B0 12       BCS $D891      yes
D87F    8E 97 02    STX $0297      0 or 1 (LOAD or SAVE)
D882    A9 40       LDA #$40
D884    8D F9 02    STA $02F9
D887    AD 4A 02    LDA $024A      file type
D88A    D0 1B       BNE $D8A7      not deleted
D88C    A9 02       LDA #$02       PRG
D88E    8D 4A 02    STA $024A      as file type
D891    AD 4A 02    LDA $024A
D894    D0 11       BNE $D8A7
D896    A5 E7       LDA $E7
D898    29 07       AND #$07       get file type and command line
D89A    8D 4A 02    STA $024A
D89D    AD 80 02    LDA $0280      track number
D8A0    D0 05       BNE $D8A7      not equal zero?
D8A2    A9 01       LDA #$01
D8A4    8D 4A 02    STA $024A      file type sequential
D8A7    AD 97 02    LDA $0297      control method
D8AA    C9 01       CMP #$01       'W'
D8AC    F0 18       BEQ $D8C6      yes
D8AE    4C 40 D9    JMP $D940

D8B1    BC 7A 02    LDY $027A,X    pointer behind second comma
D8B4    B9 00 02    LDA $0200,Y    get value
D8B7    8D 5B 02    STA $025B      record length
D8BA    AD 80 02    LDA $0280      track number
D8BD    D0 B7       BNE $D876
D8BF    A9 01       LDA #$01       'W'
D8C1    8D 97 02    STA $0297      as control method
D8C4    D0 B0       BNE $D876

D8C6    A5 E7       LDA $E7        file type
D8C8    29 80       AND #$80       isolate wildcard flag
D8CA    AA          TAX
D8CB    D0 14       BNE $D8E1      wildcard in name
D8CD    A9 20       LDA #$20
D8CF    24 E7       BIT $E7        was file closed?
D8D1    F0 06       BEQ $D8D9      yes
D8D3    20 B6 C8    JSR $C8B6      byte 0 in buffer and write block
D8D6    4C E3 D9    JMP $D9E3

D8D9    A9 80 02    LDA $0280      track number of the first block
D8DC    D0 03       BNE $D8E1      already existing
D8DE    4C E3 D9    JMP $D9E3
D8E1    AD 00 02    LDA $0200      first character from input buffer
D8E4    C9 40       CMP #$40       '@'?
D8E6    F0 0D       BEQ $D8F5      yes
D8E8    8A          TXA
D8E9    D0 05       BNE $D8F0      wildcard set?
D8EB    A9 63       LDA #$63
D8ED    4C C8 C1    JMP $C1C8      63, 'file exists'
D8F0    A9 33       LDA #$33
D8F2    4C C8 C1    JMP $C1C8      33, 'syntax error'
```

```
********************************   open a file with overwriting
D8F5   A5 E7       LDA $E7          file type
D8F7   29 07       AND #$07         isolate
D8F9   CD 4A 02    CMP $024A
D8FC   D0 67       BNE $D965        file type different?
D8FE   C9 04       CMP #$04         rel-file?
D900   F0 63       BEQ $D965        64, 'file type mismatch'
D902   20 DA DC    JSR $DCDA
D905   A5 82       LDA $82
D907   8D 70 02  , STA $0270        save channel number
D90A   A9 11       LDA #$11
D90C   20 EB D0    JSR $D0EB        open read channel
D911   AD 94 02    LDA $0294
D914   20 C8 D4    JSR $D4C8        set buffer pointer for directory
D917   A0 00       LDY #$00
D919   B1 94       LDA ($94),Y      file type
D91B   09 20       ORA #$20         set bit 5, open file
D91D   91 94       STA ($94),Y
D91F   A0 1A       LDY #$1A
D921   A5 80       LDA $80          track
D923   91 94       STA ($94),Y
D925   C8          INY
D926   A5 81       LDA $81          and sector
D928   91 94       STA ($94),Y      for open with at-sign
D92A   AE 70 02    LDX $0270        channel number
D92D   A5 D8       LDA $D8
D92F   9D 60 02    STA $0260,X      pointer to directory block
D932   A5 DD       LDA $DD
D934   9D 66 02    STA $0266,X
D937   20 3B DE    JSR $DE3B        get track and sector number
D93A   20 64 D4    JSR $D464        write block
D93D   4C EF D9    JMP $D9EF        prepare trk, sector, and drive #

D940   AD 80 02    LDA $0280        first track number
D943   D0 05       BNE $D94A        file not erased?
D945   A9 62       LDA #$62
D947   4C C8 C1    JMP $C1C8        62, 'file not found'
D94A   AD 97 02    LDA $0297        control mode
D94D   C9 03       CMP #$03         'M'
D94F   F0 0B       BEQ $D95C        yes,then no test of unclosed file
D951   A9 20       LDA #$20         bit 5
D953   24 E7       BIT $E7          test in file type
D955   F0 05       BEQ $D95C        not set, ok
D957   A9 60       LDA #$60
D959   4C C8 C1    JMP #$C1C8       60, 'write file open'
D95C   A5 E7       LDA $E7
D95E   29 07       AND #$07         isolate file type
D960   CD 4A 02    CMP $024A
D963   F0 05       BEQ $D96A
D965   A9 64       LDA #$64
D967   4C C8 C1    JMP $C1C8        64, 'file type mismatch'
D96A   A0 00       LDY #$00
D96C   8C 79 02    STY $0279
D96F   AE 97 02    LDX $0297        control mode
D972   E0 02       CPX #$02         'A', append
```

```
D974    D0 1A      BNE $D990      no
D976    C9 04      CMP #$04       rel-file?
D978    F0 EB      BEQ $D965
D97A    B1 94      LDA ($94),Y
D97C    29 4F      AND #$4F
D97E    91 94      STA ($94),Y
D980    A5 83      LDA $83
D982    48         PHA
D983    A9 11      LDA #$11
D985    85 83      STA $83        channel 17
D987    20 3B DE   JSR $DE3B      get track and sector number
D98A    20 64 D4   JSR $D464      write block
D98D    68         PLA
D98E    85 83      STA $83        get channel # back
D990    20 A0 D9   JSR $D9A0
D993    AD 97 02   LDA $0297      control mode
D996    C9 02      CMP #$02
D998    D0 55      BNE $D9EF
D99A    20 2A DA   JSR $DA2A
D99D    4C 94 C1   JMP $C194      done

D9A0    A0 13      LDA #$13
D9A2    B1 94      LDA ($94),Y    track
D9A4    8D 59 02   STA $0259
D9A7    C8         INY
D9A8    B1 94      LDA ($94),Y
D9AA    8D 5A 02   STA $025A
D9AD    C8         INY
D9AE    B1 94      LDA ($94),Y    record length
D9B0    AE 58 02   LDX $0258      last record len
D9B3    8D 58 02   STA $0258
D9B6    8A         TXA
D9B7    F0 0A      BEQ $D9C3
D9B9    CD 58 02   CMP #$0258
D9BC    F0 05      BEQ $D9C3
D9BE    A9 50      LDA #$50
D9C0    20 C8 C1   JSR $C1C8      50, 'record not present'
D9C3    AE 79 02   LDX $0279
D9C6    BD 80 02   LDA $0280,X
D9C9    85 80      STA $80        track
D9CB    BD 85 02   LDA $0285,X
D9CE    85 81      STA $81        sector
D9D0    20 46 DC   JSR $DC46
D9D3    A4 82      LDY $82
D9D5    AE 79 02   LDX $0279
D9D8    B5 D8      LDA $D8,X
D9DA    99 60 02   STA $0260,Y
D9DD    B5 DD      LDA $DD,X
D9DF    99 66 02   STA $0266,Y
D9E2    60         RTS

D9E3    A5 E2      LDA $E2        drive #
D9E5    29 01      AND #$01
D9E7    85 7F      STA $7F
D9E9    20 DA DC   JSR $DCDA
```

```
D9EC    20 E4 D6    JSR $D6E4
D9EF    A5 83       LDA $83       channel #
D9F1    C9 02       CMP #$02
D9F3    B0 11       BCS $DA06
D9F5    20 3E DE    JSR $DE3E
D9F8    A5 80       LDA $80
D9FA    85 7E       STA $7E
D9FC    A5 7F       LDA $7F
D9FE    8D 6E 02    STA $026E
DA01    A5 81       LDA $81
DA03    8D 6F 02    STA $026F
DA06    4C 99 C1    JMP $C199

******************************  check file type and control mode
DA09    BC 7A 02    LDY $027A,X   pointer in command line
DA0C    B9 00 02    LDA $0200,Y   get characters from line
DA0F    A0 04       LDY #$04
DA11    88          DEY
DA12    30 08       BMI $DA1C
DA14    D9 B2 FE    CMP $FEB2,Y   control modes 'R', 'W', 'A', 'M'
DA17    D0 F8       BNE $DA11
DA19    8C 97 02    STY $0297     save
DA1C    A0 05       LDY #$05
DA1E    88          DEY
DA1F    30 08       BMI $DA29
DA21    D9 B6 FE    CMP $FEB6,Y   file types 'D','S','P','U','L'
DA24    D0 F8       BNE $DA1E
DA26    8C 4A 02    STY $024A     save
DA29    60          RTS

******************************  preparation for Append
DA2A    20 39 CA    JSR $CA39     open channel to read, get byte
DA2D    A9 80       LDA #$80
DA2F    20 A6 DD    JSR $DDA6     last byte?
DA32    F0 F6       BEQ $DA 2A    no
DA34    20 95 DE    JSR $DE95     get track and sector number
DA37    A6 81       LDX $81       sector number
DA39    E8          INX
DA3A    8A          TXA
DA3B    D0 05       BNE $DA42     not $FF?
DA3D    20 A3 D1    JSR $D1A3     close buffer, write block
DA40    A9 02       LDA #$02
DA42    20 C8 D4    JSR $D4C8     buffer pointer to 2
DA45    A6 82       LDX $82       channel number
DA47    A9 01       LDA #$01
DA49    95 F2       STA $F2,X     set flag for WRITE
DA4B    A9 80       LDA #$80
DA4D    05 82       ORA $82
DA4F    A6 83       LDX $83
DA51    9D 2B 02    STA $022B,X   channel number in table
DA54    60          RTS

******************************  OPEN "$"
DA55    A9 0C       LDA #$0C      command number 12
DA57    8D 2A 02    STA $022A
```

```
DA5A    A9 00       LDA #$00
DA5C    AE 74 02    LDX $0274
DA5F    CA          DEX
DA60    F0 0B       BEQ $DA6D
DA62    CA          DEX
DA63    D0 21       BNE $DA86
DA65    AD 01 02    LDA $0201       second character
DA68    20 BD C3    JSR $C3BD       get drive number
DA6B    30 19       BMI $DA86       not a plain number?
DA6D    85 E2       STA $E2
DA6F    EE 77 02    INC $0277
DA72    EE 78 02    INC $0278
DA75    EE 7A 02    INC $027A
DA78    A9 80       LDA #$80
DA7A    85 E7       STA $E7         set wildcard flag
DA7C    A9 2A       LDA #$2A        '*'
DA7E    8D 00 02    STA $0200       as file name in command buffer
DA81    8D 01 02    STA $0201
DA84    D0 18       BNE $DA9E       absolute jump
DA86    20 E5 C1    JSR $C1E5       test input line to ':'
DA89    D0 05       BNE $DA90       found?
DA8B    20 DC C2    JSR $C2DC       erase flags
DA8E    A0 03       LDY #$03
DA90    88          DEY
DA91    88          DEY
DA92    8C 7A 02    STY $027A       pointer to drive no. in command
DA95    20 00 C2    JSR $C200       analyze line
DA98    20 98 C3    JSR $C398       ascertain file type
DA9B    20 20 C3    JSR $C320       get drive number
DA9E    20 CA C3    JSR $C3CA       initialize drive if necessary
DAA1    20 B7 C7    JSR $C7B7       prepare disk title
DAA4    20 9D C4    JSR $C49D       load directory
DAA7    20 9E EC    JSR $EC9E       create and prepare directory
DAAA    20 37 D1    JSR $D137       get byte from buffer
DAAD    A6 82       LDX $82         channel number
DAAF    9D 3E 02    STA $023E       byte in output register
DAB2    A4 7F       LDA $7F         drive number
DAB4    8D 8E 02    STA $028E       save as last drive number
DAB7    09 04       ORA #$04
DAB9    95 EC       STA $EC,X       PRG-flag
DABB    A9 00       LDA #$00
DABD    85 A3       STA $A3         set pointer back in input buffer
DABF    60          RTS

********************************    CLOSE-routine
DAC0    A9 00       LDA #$00
DAC2    8D F9 02    STA $02F9
DAC5    A5 83       LDA $83         secondary address
DAC7    D0 08       BNE $DAD4       not zero?
DAC9    A9 00       LDA #$00        secondary address 0, LOAD
DACB    8D 54 02    STA $0254
DACE    20 27 D2    JSR $D227       close channel
DAD1    4C DA D4    JMP $D4DA       close internal channels 17 & 18

DAD4    C9 0F       CMP #$0F        15
```

```
DAD6    F0 14       BEQ $DAEC       yes, close all channels
DAD8    20 02 DB    JSR $DB02       close file
DADB    A5 83       LDA $83         secondary address
DADD    C9 02       CMP #$02
DADF    90 F0       BCC $DAD1       smaller than 2?
DAE1    AD 6C 02    LDA $026C
DAE4    D0 03       BNE $DAE9
DAE6    4C 94 C1    JMP $C194       termination

DAE9    4C AD C1    JMP $C1AD
DAEC    A9 0E       LDA #$0E        14
DAEE    85 83       STA $83         secondary address
DAF0    20 02 DB    JSR $DB02       close file
DAF3    C6 83       DEC $83         next secondary address
DAF5    10 F9       BPL $DAF0
DAF7    AD 6C 02    LDA $026C
DAFA    D0 03       BNE $DAFF
DAFC    4C 94 C1    JMP $C194       termination
DAFF    4C AD C1    JMP $C1AD

******************************      close file
DB02    A6 83       LDX $83         secondary address
DB04    BD 2B 02    LDA $022B,X     get channel number
DB07    C9 FF       CMP #$FF        no channel associated?
DB09    D0 01       BNE $DB0C
DB0B    60          RTS             no, then done

DB0C    29 0F       AND #$0F        isolate channel number
DB0E    85 82       STA $82
DB10    20 25 D1    JSR $D125       check data type
DB13    C9 07       CMP #$07        direct access?
DB15    F0 0F       BEQ $DB26       yes
DB17    C9 04       CMP #$04        rel-file?
DB19    F0 11       BEQ $DB2C       yes
DB1B    20 07 D1    JSR $D107       channel for writing open
DB1E    B0 09       BCS $DB29       no file for writing?
DB20    20 62 DB    JSR $DB62       write last block
DB23    20 A5 DB    JSR $DBA5       write entry in dir and block
DB26    20 F4 EE    JSR $EEF4       write BAM
DB29    4C 27 D2    JMP $D227       close channel

DB2C    20 F1 DD    JSR $DDF1       get buffer number, write block
DB2F    20 1E CF    JSR $CF1E       change buffer
DB32    20 CB E1    JSR $E1CB       get last side-sector
DB35    A6 D5       LDX $D5         side-sector number
DB37    86 73       STX $73
DB39    E6 73       INC $73
DB3B    A9 00       LDA #$00
DB3D    85 70       STA $70
DB3F    85 71
DB41    A5 D6       LDA $D6
DB43    38          SEC
DB44    E9 0E       SBC #$0E        minus 14 for pointer
CB46    85 72       STA $72
DB48    20 51 DF    JSR $DF51       calculate block number of file
```

```
DB4B    A6 82       LDX $82       channel number
DB4D    A5 70       LDA $70
DB4F    95 B5       STA $B5,X     record number lo
DB51    A5 71       LDA $71
DB53    95 BB       STA $BB,X     record number hi
DB55    A9 40       LDA #$40
DB57    20 A6 DD    JSR $DDA6     bit 6 set?
DB5A    F0 03       BEQ $DB5F     no
DB5C    20 A5 DB    JSR $DBA5     enter in dirctory
DB5F    AC 27 D2    JMP $D227     close channel

******************************    write last block
DB62    A6 82       LDX $82       channel number
DB64    B5 B5       LDA $B5,X     record number lo
DB66    15 BB       ORA $BB,X     record number hi
DB68    D0 0C       BNE $DB76     not zero?
DB6A    20 E8 D4    JSR $D4E8     set buffer pointer
DB6D    C9 02       CMP #$02
DB6F    D0 05       BNE $DB76     not 2
DB71    A9 0D       LDA #$0D      CR
DB73    20 F1 CF    JSR $CFF1     in buffer
DB76    20 E8 D4    JSR $D4E8     set buffer pointer
DB79    C9 02       CMP #$02      now equal to 2?
DB7B    D0 0F       BNE $DB8C     no
DB7D    20 1E CF    JSR $CF1E     change buffer
DB80    A6 82       LDX $82       channel number
DB82    B5 B5       LDA $B5,X     record number lo
DB84    D0 02       BNE $DB88
DB86    D6 BB       DEC $BB,X     decrement block number hi
DB88    D6 B5       DEC $B5,X     and block number lo
DB8A    A9 00       LDA #$00
DB8C    38          SEC
DB8D    E9 01       SBC #$01      set pointer to end
DB8F    48          PHA
DB90    A9 00       LDA #$00
DB92    20 C8 D4    JSR $D4C8     buffer pointer to zero
DB95    20 F1 CF    JSR $CFF1     write zero in buffer
DB98    68          PLA           second byte = pointer to end
DB99    20 F1 CF    JSR $CFF1     write in buffer
DB9C    20 C7 D0    JSR $D0C7     write block to disk
DB9F    20 99 D5    JSR $D599     and verify
DBA2    4C 1E CF    JMP $CF1E     change buffer

******************************    directory entry
DBA5    A6 82       LDX $82       channel number
DBA7    8E 70 02    STX $0270     save
DBAA    A5 83       LDA $83       secondary address
DBAC    48          PHA           save
DBAD    BD 60 02    LDA $0260,X   sector number in directory
DBB0    85 81       STA $81       set
DBB2    BD 66 02    LDA $0266,X   pointer in directory
DBB5    8D 94 02    STA $0294
DBB8    B5 EC       LDA $EC,X
DBBA    29 01       AND #$01
DBBC    85 7F       STA $7F       drive number
```

```
DBBE   AD 85 FE   LDA $FE85     18, directory track
DBC1   85 80      STA $80       set
DBC3   20 93 DF   JSR $DF93     increment buffer number
DBC6   48         PHA
DBC7   85 F9      STA $F9
DBC9   20 60 D4   JSR $D460     read directory block
DBCC   A0 00      LDY #$00
DBCE   BD E0 FE   LDA $FEE0,X   buffer address
DBD1   85 87      STA $87
DBD3   AD 94 02   LDA $0294     buffer pointer
DBD6   85 86      STA $86
DBD8   B1 86      LDA ($86),Y   file type
DBDA   29 20      AND #$20      file closed?
DBDC   F0 43      BEQ $DC21     yes
DBDE   20 25 D1   JSR $D125     check file type
DBE1   C9 04      CMP #$04      rel-file?
DBE3   F0 44      BEQ $DC29     yes
DBE5   B1 86      LDA ($86),Y
DBE7   29 8F      AND #$8F      erase bits 4,5, and 6
DBE9   91 86      STA ($86),Y   in file type
DBEB   C8         INY
DBEC   B1 86      LDA ($86),Y   track number
DBEE   85 80      STA $80
DBF0   84 71      STY $71
DBF2   A0 1B      LDY #$1B
DBF4   B1 86      LDA ($86),Y   sector # of the file for
DBF6   48         PHA           overwriting
DBF7   88         DEY
DBF8   B1 86      LDA ($86),Y   track # for overwriting
DBFA   D0 0A      BNE $DC06     set?
DBFC   85 80      STA $80       set track number
DBFE   68         PLA
DBFF   85 81      STA $81       sector number
DC01   A9 67      LDA #$67
DC03   20 45 E6   JSR $E645     67, 'illegal track or sector'
DC06   48         PHA
DC07   A9 00      LDA #$00
DC09   91 86      STA ($86),Y   erase track number
DC0B   C8         INY
DC0C   91 86      STA ($86),Y   and sector number of the
DC0E   68         PLA           substitute file
DC0F   A4 71      LDY $71
DC11   91 86      STA ($86),Y
DC13   C8         INY           set track & sec # of the new file
DC14   B1 86      LDA ($86),Y
DC16   85 81      STA $81
DC18   68         PLA
DC19   91 86      STA ($86),Y
DC1B   20 7D C8   JSR $C87D     erase all files
DC1E   4C 29 DC   JMP $DC29

DC21   B1 86      LDA ($86),Y   get file type
DC23   29 0F      AND #$0F      isolate bits 0-3
DC25   09 80      ORA #$80      set bit 7 for closed file
DC27   91 86      STA ($86),Y
```

```
DC29    AE 70 02    LDX $0270      channel number
DC2C    A0 1C       LDY #$1C
DC2E    B5 B5       LDA $B5,X      block number lo
DC30    91 86       STA ($86),Y    in directory entry
DC32    C8          INY
DC33    B5 BB       LDA $BB,Y      and block number hi
DC35    91 86       STA ($86),Y    write
DC37    68          PLA            buffer number
DC38    AA          TAX
DC39    A9 90       LDA #$90       code for 'writing'
DC3B    20 90 D5    JSR $D590      write block
DC40    68          PLA
DC41    85 83       STA $83        secondary address
DC43    4C 07 D1    JMP $D107      open channel for writing

******************************     read block, layout buffer
DC46    A9 01       LDA #$01
DC48    20 E2 D1    JSR $D1E2      find channel and buffer for read
DC4B    20 B6 DC    JSR $DCB6      set pointer
DC4E    AD 4A 02    LDA $024A      file type
DC51    48          PHA            save
DC52    0A          ASL A
DC53    05 7F       ORA $7F        drive number
DC55    95 EC       STA $EC,X
DC57    20 9B D0    JSR $D09B      read block in buffer
DC5A    A6 82       LDX $82        channel number
DC5C    A5 80       LDA $80        track
DC5E    D0 05       BNE $DC65      following track?
DC60    A5 81       LDA $81        sector
DC62    9D 44 02    STA $0244,X    as end pointer
DC65    68          PLA            file type
DC66    C9 04       CMP #$04       rel-file?
DC68    D0 3F       BNE $DCA9      no
DC6A    A4 83       LDA $83        secondary address
DC6C    B9 2B 02    LDA $022B,Y    channel number
DC6F    09 40       ORA #$40
DC71    99 2B 02    STA $022B,Y    set flag for READ and WRITE
DC74    AD 58 02    LDA $0258      record length
DC77    95 C7       STA $C7,X
DC79    20 8E D2    JSR $D28E      find buffer for side-sector
DC7C    10 03       BPL $DC81      found?
DC7E    4C 0F D2    JMP $D20F      70, 'no channel'

DC81    A6 82       LDX $82        channel number
DC83    95 CD       STA $CD,X
DC85    AC 59 02    LDY $0259
DC88    84 80       STY $80        track for side-sector
DC8A    AC 5A 02    LDA $025A
DC8D    84 81       STY $81        sector for side-sector
DC8F    20 D3 D6    JSR $D6D3      transmit parameters to disk cont.
DC92    20 73 DE    JSR $DE73      read block
DC95    20 99 D5    JSR $D599      and verify
DC98    A6 82       LDX $82        channel number
DC9A    A9 02       LDA #$02
DC9C    95 C1       STA $C1,X      pointer for writing
```

```
DC9E    A9 00       LDA #$00
DCA0    20 C8 D4    JSR $D4C8      buffer pointer to zero
DCA3    20 53 E1    JSR $E153      find next record
DCA6    4C 3E DE    JMP $DE3E      get track and sector number

DCA9    20 56 D1    JSR $D156      get byte from buffer
DCAC    A6 82       LDX $82        channel number
DCAE    9D 3E 02    STA $023E,X    byte in output register
DCB1    A9 88       LDA #$88       set flag for READ
DCB3    95 F2       STA $F2,X
DCB5    60          RTS

********************************   reset pointer
DCB6    A6 82       LDX $82        channel number
DCB8    B5 A7       LDA $A7,X      buffer number
DCBA    0A          ASL A          times 2
DCBB    A8          TAY
DCBC    A9 02       LDA #$02
DCBE    99 99 00    STA $0099,Y    buffer pointer lo
DCC1    B5 AE       LDA $AE,X
DCC3    09 80       ORA #$80       set bit 7
DCC5    95 AE       STA $AE,X
DCC7    0A          ASL A
DCC8    A8          TAY
DCC9    A9 02       LDA #$02
DCCB    99 99 00    STA $0099,Y    buffer pointer lo
DCCE    A9 00       LDA #$00
DCD0    95 B5       STA $B5,X      block number lo
DCD2    95 BB       STA $BB,X      block number hi
DCD4    A9 00       LDA #$00
DCD6    9D 44 02    STA $0244,X    end pointer
DCD9    60          RTS

********************************   construct a new block
DCDA    20 A9 F1    JSR $F1A9      find free sector in BAM
DCDD    A9 01       LDA #$01
DCDF    20 DF D1    JSR $D1DF      open channel
DCE2    20 D0 D6    JSR $D6D0      transmit param to disk controller
DCE5    20 B6 DC    JSR $DCB6      reset pointer
DCE8    A6 82       LDX $82        channel number
DCEA    AD 4A 02    LDA $024A      file type
DCED    48          PHA
DCEE    0A          ASL A
DCEF    05 7F       ORA $7F        drive number
DCF1    95 EC       STA $EC,X      save as flag
DCF3    68          PLA
DCF4    C9 04       CMP #$04       rel-file?
DCF6    F0 05       BEQ $DCFD      yes
DCF8    A9 01       LDA #$01
DCFA    95 F2       STA $F2,X      set WRITE flag
DCFC    60          RTS

DCFD    A4 83       LDY $83        secondary address
DCFF    B9 2B 02    LDA $022B,Y    channel number in table
DD02    29 3F       AND #$3F       erase the top two bits
```

```
DD04    09 40      ORA #$40      set bit 6
DD06    99 2B 02   STA $022B,Y   READ and WRITE flag
DD09    AD 58 02   LDA $0258     record length
DD0C    95 C7      STA $C7,X     in table
DD0E    20 8E D2   JSR $D28E     find buffer
DD11    10 03      BPL $DD16     found?
DD13    4C 0F D2   JMP $D20F     70, 'no channel'

DD16    A6 82      LDX $82       channel number
DD18    95 CD      STA $CD,X     buffer number for side-sector
DD1A    20 C1 DE   JSR $DEC1     erase buffer
DD1D    20 1E F1   JSR $F11E     find free block in BAM
DD20    A5 80      LDA $80       track
DD22    8D 59 02   STA $0259     for side-sector
DD25    A5 81      LDA $81       sector
DD27    8D 5A 02   STA $025A     for side-sector
DD2A    A6 82      LDX $82       channel number
DD2C    B5 CD      LDA $CD,X     buffer number
DD2E    20 D3 D6   JSR $D6D3     transmit param to disk controller
DD31    A9 00      LDA #$00
DD33    20 E9 DE   JSR $DEE9     buffer pointer to zero
DD36    A9 00      LDA #$00
DD38    20 8D DD   JSR $DD8D
DD3B    A9 11      LDA #$11      17
DD3D    20 8D DD   JSR $DD8D     as end pointer in buffer
DD40    A9 00      LDA #$00      zero
DD42    20 8D DD   JSR $DD8D     as side-sector number in buffer
DD45    AD 58 02   LDA $0258     record length
DD48    20 8D DD   JSR $DD8D     in buffer
DD4B    A5 80      LDA $80       track number of this block
DD4D    20 8D DD   JSR $DD8D     in buffer
DD50    A5 81      LDA $81       sector number
DD52    20 8D DD   JSR $DD8D     in buffer
DD55    A9 10      LDA #$10      16
DD57    20 E9 DE   JSR $DEE9     buffer pointer to 16
DD5A    20 3E DE   JSR $DE3E     get track and sector number
DD5D    A5 80      LDA $80       track # of the first data block
DD5F    20 8D DD   JSR $DD8D     in buffer
DD62    A5 81      LDA $81       sector # of the first data block
DD64    20 8D DD   JSR $DD8D     in buffer
DD67    20 6C DE   JSR $DE6C     write block to disk
DD6A    20 99 D5   JSR $D599     and check
DD6D    A9 02      LDA #$02
DD6F    20 C8 D4   JSR $D4C8     buffer pointer to 2
DD72    A6 82      LDX $82       channel number
DD74    38         SEC
DD75    A9 00      LDA #$00
DD77    F5 C7      SBC $C7,X     record length
DD79    95 C1      STA $C1,X     pointer for writing
DD7B    20 E2 E2   JSR $E2E2     erase buffer
DD7E    20 19 DE   JSR $DE19     write link bytes in buffer
DD81    20 5E DE   JSR $DE5E     write block to disk
DD84    20 99 D5   JSR $D599     and check
DD87    20 F4 EE   JSR $EEF4     write BAM
DD8A    4C 98 DC   JMP $DC98     and done
```

```
********************************     write byte in side-sector block
DD8D    48          PHA              save byte
DD8E    A6 82       LDX $82          channel number
DD90    B5 CD       LDA $CD,X        buffer # of the side-sector
DD92    4C FD CF    JMP $CFFD        write byte in buffer

********************************     manipulate flags
DD95    90 06       BCC $DD9D
DD97    A6 82       LDX $82          channel number
DD99    15 EC       ORA $EC,X        set flag
DD9B    D0 06       BNE $DDA3
DD9D    A6 82       LDX $82          channel number
DD9F    49 FF       EOR #$FF
DDA1    35 EC       AND $EC,X        erase flag
DDA3    95 EC       STA $EC,X
DDA5    60          RTS
DDA6    A6 82       LDX $82          channel number
DDA8    35 EC       AND $EC,X        test flag
DDAA    60          RTS

********************************     check command code for writing
DDAB    20 93 DF    JSR $DF93        get buffer number
DDAE    AA          TAX
DDAF    BD 5B 02    LDA $025B,X
DDB2    29 F0       AND #$F0         isolate command code
DDB4    C9 90       CMP #$90         code for writing?
DDB6    60          RTS

********************************
DDB7    A2 00       LDX #$00
DDB9    86 71       STX $71          counter for secondary address
DDBB    BD 2B 02    LDA $022B,X      get channel number from table
DDBE    C9 FF       CMP #$FF
DDC0    D0 08       BNE $DDCA        file open?
DDC2    A6 71       LDX $71
DDC4    E8          INX              increment counter
DDC5    E0 10       CPX #$10         smaller than 16?
DDC7    90 F0       BCC $DDB9
DDC9    60          RTS

DDCA    86 71       STX $71
DDCC    29 3F       AND #$3F         isolate channel number
DDCE    A8          TAY
DDCF    B9 EC 00    LDA $00EC,Y
DDD2    29 01       AND #$01         isolate drive number
DDD4    85 70       STA $70
DDD6    AE 53 02    LDX $0253
DDD9    B5 E2       LDA $E2,X
DDDB    29 01       AND #$01         isolate drive number
DDDD    C5 70       CMP $70          same drive?
DDDF    D0 E1       BNE $DDC2        no
DDE1    B9 60 02    LDA $0260,Y      sector number in directory
DDE4    D5 D8       CMP $D8,X        same as file?
DDE6    D0 DA       BNE $DDC2        no
```

```
DDE8    B9 66 02    LDA $0266,Y
DDEB    D5 DD       CMP $DD,X       pointer same?
DDED    D0 D3       BNE $DDC2       no
DDEF    18          CLC
DDF0    60          RTS

********************************    write a block of a rel-file
DDF1    20 9E DF    JSR $DF9E       get buffer number
DDF4    50 06       BVC $DDFC       no rel-file?
DDF6    20 5E DE    JSR $DE5E       write block
DDF9    20 99 D5    JSR $D599       and verify
DDFC    60          RTS

********************************    write bytes for following track
DDFD    20 2B DE    JSR $DE2B       set buffer pointer
DE00    A5 80       LDA $80         track number
DE02    91 94       STA ($94),Y     in buffer
DE04    C8          INY
DE05    A5 81       LDA $81         sector number
DE07    91 94       STA ($94),Y     in buffer
DE09    4C 05 E1    JMP $E105       set rel-flag

********************************    get following track and sector #
DE0C    20 2B DE    JSR $DE2B       set buffer pointer
DE0F    B1 94       LDA ($94),Y     following track number
DE11    85 80       STA $80
DE13    C8          INY
DE14    B1 94       LDA ($94),Y     and get sector number
DE16    85 81       STA $81
DE18    RTS

********************************    following track for last block
DE19    20 2B DE    JSR $DE2B       set buffer pointer
DE1C    A9 00       LDA #$00        zero
DE1E    91 94       STA ($94),Y     as track number
DE20    C8          INY
DE21    A6 82       LDX $82         channel number
DE23    B5 C1       LDA $C1,X       pointer in block
DE25    AA          TAX
DE26    CA          DEX             minus 1
DE27    8A          TXA
DE28    91 94       STA ($94),Y     as pointer in block
DE2A    60          RTS

********************************    buffer pointer to zero
DE2B    20 93 DF    JSR $DF93       get buffer number
DE2E    0A          ASL A           times 2
DE2F    AA          TAX
DE30    B5 9A       LDA $9A,X       buffer pointer hi
DE32    85 95       STA $95
DE34    A9 00       LDA #$00
DE36    85 94       STA $94         buffer pointer lo
DE38    A0 00       LDY #$00
DE3A    60          RTS
```

```
********************************     get track and sector
DE3B    20 EB D0    JSR $D0EB        get channel number
DE3E    20 93 DF    JSR $DF93        get buffer number
DE41    85 F9       STA $F9          save
DE43    0A          ASL A            times 2
DE44    A8          TAY
DE45    B9 06 00    LDA $0006,Y      get track
DE48    85 80       STA $80
DE4A    B9 07 00    LDA $0007,Y      and sector # from disk controller
DE4D    85 81       STA $81
DE4F    60          RTS

********************************
DE50    A9 90       LDA #$90         command code for writing
DE52    8D 4D 02    STA $024D
DE55    D0 28       BNE $DE7F

DE57    A9 80       LDA #$80         command code for reading
DE59    8D 4D 02    STA $024D
DE5C    D0 21       BNE $DE7F
DE5E    A9 90       LDA #$90         command code for writing
DE60    8D 4D 02    STA $024D
DE63    D0 26       BNE $DE8B

DE65    A9 80       LDA #$80         command code for reading
DE67    8D 4D 02    STA $024D
DE6A    D0 1F       BNE $DE8B

DE6C    A9 90       LDA #$90         command code for writing
DE6E    8D 4D 02    STA $024D
DE71    D0 02       BNE $DE75

DE73    A9 80       LDA #$80         command code for reading
DE75    8D 4D 02    STA $024D
DE78    A6 82       LDX $82          channel number
DE7A    B5 CD       LDA $CD,X        side-sector buffer number
DE7C    AA          TAX
DE7D    10 13       BPL $DE92        buffer associated?
DE7F    20 D0 D6    JSR $D6D0        generate header for disk cont.
DE82    20 93 DF    JSR $DF93        get buffer number
DE85    AA          TAX
DE86    A5 7F       LDA $7F          drive number
DE88    9D 5B 02    STA $025B,X
DE8B    20 15 E1    JSR $E115        buffer number
DE8E    20 93 DF    JSR $DF93        get buffer number
DE91    AA          TAX
DE92    4C 06 D5    JMP $D506        write block

********************************     get following track & sector from
DE95    A9 00       LDA #$00         buffer
DE97    20 C8 D4    JSR $D4C8        buffer pointer to zero
DE9A    20 37 D1    JSR $D137        get byte
DE9D    85 80       STA $80          save as track
DE9F    20 37 D1    JSR $D137        get byte
DEA2    85 81       STA $81          as sector
```

```
DEA4    60          RTS

******************************  copy buffer contents
DEA5    48          PHA
DEA6    A9 00       LDA #$00
DEA8    85 6F       STA $6F
DEAA    85 71       STA $71
DEAC    B9 E0 FE    LDA $FEE0,Y    buffer address Y, hi
DEAF    85 70       STA $70
DEB1    BD E0 FE    LDA $FEE0,X    buffer address X, hi
DEB4    85 72       STA $72
DEB6    68          PLA
DEB7    A8          TAY
DEB8    88          DEY
DEB9    B1 6F       LDA ($6F),Y    copy contents of buffer Y
DEBB    91 71       STA ($71),Y    to buffer X
DEBD    88          DEY
DEBE    10 F9       BPL $DEB9
DEC0    60          RTS

******************************  erase buffer Y
DEC1    A8          TAY            buffer number
DEC2    B9 E0 FE    LDA $FEE0,Y    get hi-address
DEC5    85 70       STA $70
DEC7    A9 00       LDA #$00       lo-address
DEC9    85 6F       STA $6F
DECB    A8          TAY
DECC    91 6F       STA ($6F),Y    erase buffer
DECE    C8          INY
DECF    D0 FB       BNE $DECC
DED1    60          RTS

******************************  get side-sector number
DED2    A9 00       LDA #$00
DED4    20 DC DE    JSR $DEDC      buffer pointer to zero
DED7    A0 02       LDY #$02
DED9    B1 94       LDA ($94),Y    byte 2 contains the side-sector #
DEDB    60          RTS

******************************  set buffer ptr to side-sector
DEDC    85 94       STA $94        pointer lo
DEDE    A6 82       LDX $82        channel number
DEE0    B5 CD       LDA $CD,X      buffer number
DEE2    AA          TAX
DEE3    BD E0 FE    LDA $FEE0,X    buffer address hi
DEE6    85 95       STA $95        set
DEE8    60          RTS

******************************  buffer pointer for side-sector
DEE9    48          PHA            pointer in side-sector
DEEA    20 DC DE    JSR $DEDC      set buffer pointer
DEED    48          PHA
DEEE    8A          TXA            buffer number
DEEF    0A          ASL A          times 2
DEF0    AA          TAX
```

```
DEF1    68          PLA           buffer pointer hi
DEF2    95 9A       STA $9A,X
DEF4    68          PLA           buffer pointer lo
DEF5    95 99       STA $99,X
DEF7    60          RTS

******************************    get side-sector and buffer ptr
DEF8    20 66 DF    JSR $DF66     is side-sector in buffer
DEFB    30 0E       BMI $DF0B     no
DEFD    50 13       BVC $DF12     ok
DEFF    A6 82       LDX $82       channel number
DF01    B5 CD       LDA $CD,X     buffer number
DF03    20 1B DF    JSR $DF1B     read side-sector
DF06    20 66 DF    JSR $DF66     and check if in buffer
DF09    10 07       BPL $DF12     yes?
DF0B    20 CB E1    JSR $E1CB     get last side-sector
DF0E    2C CE FE    BIT $FECE     set V bit
DF11    60          RTS
DF12    A5 D6       LDA $D6       side-sector end pointer
DF14    20 E9 DE    JSR $DEE9     set pointer in side-sector
DF17    2C CD DE    BIT $FECD     erase V bit
DF1A    60          RTS

******************************    read side-sector
DF1B    85 F9       STA $F9       buffer number
DF1D    A9 80       LDA #$80      command code for reading
DF1F    D0 04       BNE $DF25

******************************    write side-sector
DF21    85 F9       STA $F9       buffer number
DF23    A9 90       LDA #$90      command code for writing
DF25    48          PHA
DF26    B5 EC       LDA $EC,X
DF28    29 01       AND #$01      isolate drive number
DF2A    85 7F       STA $7F
DF2C    68          PLA
DF2D    05 7F       ORA $7F       command code plus drive number
DF2F    8D 4D 02    STA $024D     save
DF32    B1 94       LDA ($94),Y   track number
DF34    85 80       STA $80
DF36    C8          INY
DF37    B1 94       LDA ($94),Y   sector number
DF39    85 81       STA $81
DF3B    A5 F9       LDA $F9       buffer number
DF3D    20 D3 D6    JSR $D6D3     transmit param to disk controller
DF40    A6 F9       LDX $F9       buffer number
DF42    4C 93 D5    JMP $D593     tranmit cmd to disk controller

******************************    set buffer pointer in side-sector
DF45    A6 82       LDX $82       channel number
DF47    B5 CD       LDA $CD,X     buffer number
DF49    4C EB D4    JMP $D4EB     set buffer pointer

******************************    calculate block # of a rel-file
DF4C    A9 78       LDA #$78      120 block ptrs per side-sector
```

```
DF4E    20 5C DF   JSR $DF5C      add to $70/$71
DF51    CA         DEX            side-sector number
DF52    10 F8      BPL $DF4C      next side-sector?
DF54    A5 72      LDA $72        pointer value in last block
DF56    4A         LSR A          divided by 2
DF57    20 5C DF   JSR $DF5C      add to previous sum
DF5A    A5 73      LDA $73        number of the side-sector block
DF5C    18         CLC
DF5D    65 70      ADC $70
DF5F    85 70      STA $70        add
DF61    90 02      BCC $DF65
DF63    E6 71      INC $71
DF65    60         RTS

********************************  verify side-sector in buffer
DF66    20 D2 DE   JSR $DED2      get side-sector number
DF69    C5 D5      CMP $D5        = number of necessary block?
DF6B    D0 0E      BNE $DF7B      no
DF6D    A4 D6      LDY $D6        pointer in side-sector
DF6F    B1 94      LDA ($94),Y    track number
DF71    F0 04      BEQ $DF77
DF73    2C CD FE   BIT $FECD      erase bits
DF76    60         RTS
DF77    2C CF FE   BIT $FECF      set N-bit
DF7A    60         RTS

DF7B    A5 D5      LDA $D5        side-sector number
DF7D    C9 06      CMP #$06       6 or greater?
DF7F    B0 0A      BCS $DF8B      yes
DF81    0A         ASL A
DF82    A8         TAY
DF83    A9 04      LDA #$04
DF85    85 94      STA $94
DF87    B1 94      LDA ($94),Y    track number
DF89    D0 04      BNE $DF8F
DF8B    2C D0 FE   BIT $FED0      set N and V bits
DF8E    60         RTS

DF8F    2C CE FE   BIT $FECE      set V bit
DF92    60         RTS

********************************  get buffer number
DF93    A6 82      LDX $82        channel number
DF95    B5 A7      LDA $A7,X      buffer number
DF97    10 02      BPL $DF9B
DF99    B5 AE      LDA $AE,X      buffer number from second table
DF9B    29 BF      AND #$BF       erase V bit
DF9D    60         RTS

DF9E    A6 82      LDX $82        channel number
DFA0    8E 57 02   STX $0257      save
DFA3    B5 A7      LDA $A7,X      get buffer number
DFA5    10 09      BPL $DFB0      buffer allocated
DFA7    8A         TXA
DFA8    18         CLC
```

```
DFA9    69 07       ADC #$07         increment number by 7
DFAB    8D 57 02    STA $0257        and save
DFAE    B5 AE       LDA $AE,X        buffer number from table 2
DFB0    85 70       STA $70
DFB2    29 1F       AND #$1F         erase the highest 3 bits
DFB4    24 70       BIT $70
DFB6    60          RTS

DFB7    AD 82       LDX $82          channel number
DFB9    B5 A7       LDA $A7,X        buffer number
DFBB    30 02       BMI $DFBF        buffer free?
DFBD    B5 AE       LDA $AE,X        buffer number from table 2
DFBF    C9 FF       CMP #$FF         free?
DFC1    60          RTS

DFC2    A6 82       LDX $82
DFC4    09 80       ORA #$80
DFC6    B4 A7       LDY $A7,X
DFC8    10 03       BPL $DFCD
DFCA    95 A7       STA $A7,X
DFCC    60          RTS
DFCD    95 AE       STA $AE,X
DFCF    60          RTS

********************************  get next record in rel-file
DFD0    A9 20       LDA #$20
DFD2    20 9D DD    JSR $DD9D        erase bit 5
DFD5    A9 80       LDA #$80
DFD7    20 A6 DD    JSR $DDA6        test bit 7
DFDA    D0 41       BNE $E01D        set?
DFDC    A6 82       LDX $82          channel number
DFDE    F6 B5       INC $B5,X        increment record number
DFE0    D0 02       BNE $DFE4
DFE2    F6 BB       INC $BB,X        record number hi
DFE4    A6 82       LDX $82          channel number
DFE6    B5 C1       LDA $C1,X        write pointer
DFE8    F0 2E       BEQ $E018        zero?
DFEA    20 E8 D4    JSR $D4E8        set buffer pointer
DFED    A6 82       LDX $82          channel number
DFEF    D5 C1       CMP $C1,X        buffer ptr smaller than write ptr
DFF1    90 03       BCC $DFF6        yes
DFF3    20 3C E0    JSR $E03C        write block, read next block
DFF6    A6 82       LDX $82          channel number
DFF8    B5 C1       LDA $C1,X        write pointer
DFFA    20 C8 D4    JSR $D4C8        set buffer pointer = write ptr
DFFD    A1 99       LDA ($99),X      byte from buffer
DFFF    85 85       STA $85          put in output register
E001    A9 20       LDA #$20
E003    20 9D DD    JSR $DD9D        erase bit 5
E006    20 04 E3    JSR $E304        add record length to write ptr
E009    48          PHA              and save
E00A    90 28       BCC $E034        not yet in last block?
E00C    A9 00       LDA #$00
E00E    20 F6 D4    JSR $D4F6        get track number
E011    D0 21       BNE $E034        does block exist?
```

```
E013    68          PLA             pointer
E014    C9 02       CMP #$02        = 2
E016    F0 12       BEQ $E02A       yes
E018    A9 80       LDA #$80
E01A    20 97 DD    JSR $DD97       set bit 7
E01D    20 2F D1    JSR $D12F       get byte from buffer
E020    B5 99       LDA $99,X       buffer pointer
E022    99 44 02    STA $0244,Y     as end pointer
E025    A9 0D       LDA #$0D        CR
E027    85 85       STA $85         in output register
E029    60          RTS

E02A    20 35 E0    JSR $E035
E02D    A6 82       LDX $82         channel number
E02F    A9 00       LDA #$00
E031    95 C1       STA $C1,X       write pointer to zero
E033    60          RTS

E034    68          PLA
E035    A6 82       LDX $82         channel number
E037    95 C1       STA $C1,X       set write pointer
E039    4C 6E E1    JMP $E16E

******************************      write block and read next block
E03C    20 D3 D1    JSR $D1D3       get drive number
E03F    20 95 DE    JSR $DE95       get track and sector number
E042    20 9E DF    JSR $DF9E       get buffer number
E045    50 16       BVC $E05D       no rel-file?
E047    20 5E DE    JSR $DE5E       write block
E04A    20 1E CF    JSR $CF1E       change buffer
E04D    A9 02       LDA #$02
E04F    20 C8 D4    JSR $D4C8       buffer pointer to 2
E052    20 AB DD    JSR $DDAB       command code for writing?
E055    D0 24       BNE $E078       no
E057    20 57 DE    JSR $DE57       read block
E05A    4C 99 D5    JMP $D599       and verify

E05D    20 1E CF    JSR $CF1E       change buffer
E060    20 AB DD    JSR $DDAB       command code for writing?
E063    D0 06       BNE $E068       no
E065    20 57 DE    JSR $DE57       read block
E068    20 99 D5    JSR $D599       and verify
E06B    20 95 DE    JSR $DE95       get track and sector number
E06E    A5 80       LDA $80         track
E070    F0 09       BEQ $E07B       no following track
E072    20 1E CF    JSR $CF1E       change buffer
E075    20 57 DE    JSR $DE57       read block
E078    20 1E CF    JSR $CF1E       change buffer
E07B    60          RTS

******************************      write a byte in a record
E07C    20 05 E1    JSR $E105
E07F    20 93 DF    JSR $DF93       get buffer number
E082    0A          ASL A           times 2
E083    AA          TAX
```

```
E084    A5 85       LDA $85         data byte
E086    81 99       STA ($99,X)     write in buffer
E088    B4 99       LDY $99,X       buffer pointer
E08A    C8          INY             increment
E08B    D0 09       BNE $E096       not equal zero?
E08D    A4 82       LDY $82         channel number
E08F    B9 C1 00    LDA $00C1,Y     write pointer
E092    F0 0A       BEQ $E09E       equal zero?
E094    A0 02       LDY #$02        buffer pointer to 2
E096    98          TYA
E097    A5 82       LDY $82         channel number
E099    D9 C1 00    CMP $00C1,Y     buffer pointer = write pointer?
E09C    D0 05       BNE $E043       no
E09E    A9 20       LDA #$20
E0A0    4C 97 DD    JMP $DD97       set bit 5

E0A3    F6 99       INC $99,X       increment buffer pointer
E0A5    D0 03       BNE $E0AA       not zero?
E0A7    20 3C E0    JSR $E03C       else write block, read next one
E0AA    60          RTS

********************************    write byte in rel-file
E0AB    A9 A0       LDA #$A0
E0AD    20 A6 DD    JSR $DDA6       test bits 6 & 7
E0B0    D0 27       BNE $E0D9       set?
E0B2    A5 85       LDA $85         data byte
E0B4    20 7C E0    JSR $E07C       write in record
E0B7    A5 F8       LDA $F8         end?
E0B9    F0 0D       BEQ $E0C8       yes
E0BB    60          RTS

E0BC    A9 20       LDA #$20
E0BE    20 A6 DD    JSR $DDA6       test bit 5
E0C1    F0 05       BEQ $E0C8       not set
E0C3    A9 51       LDA #$51        51, 'overflow in record'
E0C5    8D 6C 02    STA $026C       set error flag
E0C8    20 F3 E0    JSR $E0F3       fill remainder with zeroes
E0CB    20 53 E1    JSR $E153
E0CE    AD 6C 02    LDA $026C       error flag set?
E0D1    F0 03       BEQ $E0D6       no
E0D3    4C C8 C1    JMP $C1C8       set error message

E0D6    4C BC E6    JMP $E6BC       error free execution
E0D9    29 80       AND #$80        bit 7 set?
E0DB    D0 05       BNE $E0E2       yes
E0DD    A5 F8       LDA $F8
E0DF    F0 DB       BEQ $E0BC       end?
E0E1    60          RTS

E0E2    A5 85       LDA $85         data byte
E0E4    48          PHA
E0E5    20 1C E3    JSR $E31C       expand side-sector
E0E8    68          PLA
E0E9    85 85       STA $85
E0EB    A9 80       LDA #$80
```

```
E0ED    20 9D DD    JSR $DD9D      erase bit 7
E0F0    4C B2 E0    JMP $E0B2      write byte in file

******************************     fill record with zeroes
E0F3    A9 20       LDA #$20
E0F5    20 A6 DD    JSR $DDA6      test bit 5
E0F8    D0 0A       BNE $E104      set?
E0FA    A9 00       LDA #$00
E0FC    85 85       STA $85        zero as data byte
E0FE    20 7C E0    JSR $E07C      write in record
E101    4C F3 E0    JMP $E0F3      until record full

E104    60          RTS

******************************     write buffer number in table
E105    A9 40       LDA #$40
E107    20 97 DD    JSR $DD97      set bit 6
E10A    20 9E DF    JSR $DF9E      get buffer number
E10D    09 40       ORA #$40       set bit 6
E10F    AE 57 02    LDX $0257      channel number + 7
E112    95 A7       STA $A7,X      write in table
E114    60          RTS

E115    20 9E DF    JSR $DF9E      get buffer number
E118    29 BF       AND #$BF       erase bit 6
E11A    AE 57 02    LDX $0257      channel number
E11D    95 A7       STA $A7,X      write in table
E11F    60          RTS

******************************     get byte from rel-file
E120    A9 80       LDA #$80
E122    20 A6 DD    JSR $DDA6      test bit 7
E125    D0 37       BNE $E15E      set?
E127    20 2F D1    JSR $D12F      get byte from buffer
E12A    B5 99       LDA $99,X      buffer pointer
E12C    D9 44 02    CMP $0244,Y    compare to end pointer
E12F    F0 22       BEQ $E135      equal?
E131    F6 99       INC $99,X      increment buffer pointer
E133    D0 06       BNE $E13B      not zero?
E135    20 3C E0    JSR $E03C      write block, read next one
E138    20 2F D1    JSR $D12F      get byte from buffer
E13B    A1 99       LDA ($99,X)
E13D    99 3E 02    STA $023E,Y    in output register
E140    A9 89       LDA #$89
E142    99 F2 00    STA $00F2,Y    set READ and WRITE flag
E145    B5 99       LDA $99,Y      buffer pointer
E147    D9 44 02    CMP $0244,Y    compare to end pointer
E14A    F0 01       BEQ $E14D      same?
E14C    60          RTS

E14D    A9 81       LDA #$81
E14F    99 F2 00    STA $00F2,Y    set flag for end
E152    60          RTS

E153    20 D0 DF    JSR $DFD0      find next record
```

```
E156    20 2F D1    JSR $D12F     get buffer and channel number
E159    A5 85       LDA $85       data byte
E15B    4C 3D E1    JMP $E13D     into output register

E15E    A6 82       LDX $82       channel number
E160    A9 0D       LDA #$0D      CR
E162    9D 3E 02    STA $023E,X   into output register
E165    A9 81       LDA #$81
E167    95 F2       STA $F2,X     set flag for end
E169    A9 50       LDA #$50
E16B    20 C8 C1    JSR $C1C8     50, 'record not present'

E16E    A6 82       LDX $82       channel number
E170    B5 C1       LDA $C1,X     write pointer
E172    85 87       STA $87       save
E174    C6 87       DEC $87
E176    C9 02       CMP #$02      equal 2?
E178    D0 04       BNE $E17E     no
E17A    A9 FF       LDA #$FF
E17C    85 87       STA $87
E17E    B5 C7       LDA $C7,X     record length
E180    85 88       STA $88
E182    20 E8 D4    JSR $D4E8     set buffer pointer
E185    A6 82       LDX $82       channel number
E187    C5 87       CMP $87       buffer pointer > write pointer?
E189    90 19       BCC $E1A4
E18B    F0 17       BEQ $E1A4     no
E18D    20 1E CF    JSR $CF1E     change buffer
E190    20 B2 E1    JSR $E1B2
E193    90 08       BCC $E19D
E195    A6 82       LDX $82       channel number
E197    9D 44 02    STA $0244,X
E19A    4C 1E CF    JMP $CF1E     change buffer

E19D    20 1E CF    JSR $CF1E     change buffer
E1A0    A9 FF       LDA #$FF
E1A2    85 87       STA $87
E1A4    20 B2 E1    JSR $E1B2
E1A7    B0 03       BCS $E1AC
E1A9    20 E8 D4    JSR $D4E8     set buffer pointer
E1AC    A6 82       LDX $82       channel number
E1AE    9D 44 02    STA $0244,X   end pointer
E1B1    60          RTS

E1B2    20 2B DE    JSR $DE2B     buffer pointer to zero
E1B5    A4 87       LDY $87
E1B7    B1 94       LDA ($94),Y   byte from buffer
E1B9    D0 0D       BNE $E1C8     not zero?
E1BB    88          DEY
E1BC    C0 02       CPY #$02
E1BE    90 04       BCC $E1C4
E1C0    C6 88       DEC $88
E1C2    D0 F3       BNE $E1B7
E1C4    C6 88       DEC $88
E1C6    18          CLC
```

```
E1C7    60          RTS

E1C8    98          TYA
E1C9    38          SEC
E1CA    60          RTS

********************************    get last side-sector
E1CB    20 D2 DE    JSR $DED2       get number of the side-sector
E1CE    85 D5       STA $D5         save
E1D0    A9 04       LDA #$04
E1D2    85 94       STA $94         pointer to side-sectors
E1D4    A0 0A       LDY #$0A
E1D6    D0 04       BNE $E1DC

E1D8    88          DEY
E1D9    88          DEY
E1DA    30 26       BMI $E202
E1DC    B1 94       LDA ($94),Y     track # of the previous block
E1DE    F0 F8       BEQ $E1D8
E1E0    98          TYA
E1E1    4A          LSR A           divide by 2
E1E2    C5 D5       CMP $D5         = number of the actual block?
E1E4    F0 09       BEQ $E1EF       yes
E1E6    85 D5       STA $D5         else save all numbers
E1E8    A6 82       LDX $82         channel number
E1EA    B5 CD       LDA $CD,X       buffer number
E1EC    20 1B DF    JSR $DF1B       read block
E1EF    A0 00       LDY #$00
E1F1    84 94       STY $94         buffer pointer
E1F3    B1 94       LDA ($94),Y     track number
E1F5    D0 0B       BNE $E202       another block?
E1F7    C8          INY
E1F8    B1 94       LDA ($94),Y     sector number = end pointer
E1FA    A8          TAY
E1FB    88          DEY
E1FC    84 D6       STY $D6         save end pointer
E1FE    98          TYA
E1FF    4C E9 DE    JMP $DEE9       set buffer pointer

E202    A9 67       #$67
E204    20 45 E6    JSR $E645       67, 'illegal track or sector'

********************************    P-command, 'Record'
E207    20 B3 C2    JSR $C2B3       verify lines
E20A    AD 01 02    LDA $0201       secondary address
E20D    85 83       STA $83
E20F    20 EB D0    JSR $D0EB       find channel number
E212    90 05       BCC $E219       found?
E214    A9 70       LDA #$70
E216    20 C8 C1    JSR $C1C8       70, 'no block'

E219    A9 A0       LDA #$A0
E21B    20 9D DD    JSR $DD9D       erase bits 6 & 7
E21E    20 25 D1    JSR $D125       verify if 'REL'-file
E221    F0 05       BEQ $E228       yes
```

```
E223    A9 64       LDA #$64
E225    20 C8 C1    JSR $C1C8      64, 'file type mismatch'

E228    B5 EC       LDA $EC,X
E22A    29 01       AND #$01
E22C    85 7F       STA $7F        drive number
E22E    AD 02 02    LDA $0202      record number lo
E231    95 B5       STA $B5,X
E233    AD 03 02    LDA $0203      record number hi
E236    95 BB       STA $BB,X
E238    A6 82       LDA $82        channel number
E23A    A9 89       LDA #$89
E23C    95 F2       STA $F2,X      READ and WRITE flag
E23E    AD 04 02    LDA $0204      byte-pointer
E241    F0 10       BEQ $E253      zero?
E243    38          SEC
E244    E9 01       SBC #$01
E246    F0 0B       BEQ $E253
E248    D5 C7       CMP $C7,X      compare with record length
E24A    90 07       BCC $E253
E24C    A9 51       LDA #$51
E24E    8D 6C 02    STA $026C      51, 'overflow in record'
E251    A9 00       LDA #$00
E253    85 D4       STA $D4
E255    20 0E CE    JSR $CE0E      calculate pointer in rel-file
E258    20 F8 DE    JSR $DEF8      and read appropriate side-sector
E25B    50 08       BVC $E265      does block exist?
E25D    A9 80       LDA #$80
E25F    20 97 DD    JSR $DD97      set bit 7
E262    4C 5E E1    JMP $E15E      and 50, 'record not present'

E265    20 75 E2    JSR $E275
E268    A9 80       LDA #$80
E26A    20 A6 DD    JSR $DDA6      test bit 7
E26D    F0 03       BEQ $E272      not set
E26F    4C 5E E1    JMP $E15E      50, 'record not present'
E272    4C 94 C1    JMP $C194      done

E275    20 9C E2    JSR $E29C
E278    A5 D7       LDA $D7        pointer in rel-file
E27A    20 C8 D4    JSR $D4C8      set buffer pointer
E27D    A6 82       LDX $82        channel number
E27F    B5 C7       LDA $C7,X      record length
E281    38          SEC
E282    E5 D4       SBC $D4        minus position
E284    B0 03       BCS $E289      positive?
E286    4C 02 E2    JMP $E202      67, 'illegal track or sector'

E289    18          CLC
E28A    65 D7       ADC $D7        add pointer in data block
E28C    90 03       BCC $E291      no overflow
E28E    69 01       ADC #$01       plus 2
E290    38          SEC
E291    20 09 E0    JSR $E009      set pointer
E294    4C 38 E1    JMP $E138      get byte from buffer
```

```
E297    A9 51       LDA #$51
E299    20 C8 C1    JSR $C1C8       51, 'overflow in record'

E29C    A5 94       LDA $94         buffer pointer lo
E29E    85 89       STA $89
E2A0    A5 95       LDA $95         buffer pointer hi
E2A2    85 8A       STA $8A
E2A4    20 D0 E2    JSR $E2D0       compare track and sector
E2A7    D0 01       BNE $E2AA       not equal?
E2A9    60          RTS

E2AA    20 F1 DD    JSR $DDF1
E2AD    20 0C DE    JSR $DE0C
E2B0    A5 80       LDA $80         track
E2B2    F0 0E       BEQ $E2C2       no block following?
E2B4    20 D3 E2    JSR $E2D3       compare track and sector number
E2B7    D0 06       BNE $E2BF       not equal?
E2B9    20 1E CF    JSR $CF1E       change buffer
E2BC    4C DA D2    JMP $D2DA
E2BF    20 DA D2    JSR $D2DA
E2C2    A0 00       LDY #$00
E2C4    B1 89       LDA ($89),Y     track
E2C6    85 80       STA $80
E2C8    C8          INY
E2C9    B1 89       LDA ($89),Y     and sector of the next block
E2CB    85 81       STA $81
E2CD    4C AF D0    JMP $D0AF       read block

E2D0    20 3E DE    JSR $DE3E
E2D3    A0 00       LDY #$00
E2D5    B1 89       LDA ($89),Y     track number
E2D7    C5 80       CMP $80         compare
E2D9    F0 01       BEQ $E2DC
E2DB    60          RTS
E2DC    C8          INY
E2DD    B1 89       LDA ($89),Y     sector number
E2DF    C5 81       CMP $81         compare
E2E1    60          RTS

******************************      subdivide records in data block
E2E2    20 2B DE    JSR $DE2B       set buffer pointer
E2E5    A0 02       LDY #$02
E2E7    A9 00       LDA #$00
E2E9    91 94       STA ($94),Y     erase buffer
E2EB    C8          INY
E2EC    D0 FB       BNE $E2E9
E2EE    20 04 E3    JSR $E304       set pointer to next record
E2F1    95 C1       STA $C1,X
E2F3    A8          TAY
E2F4    A9 FF       LDA #$FF
E2F6    91 94       STA ($94),Y     $FF as 1st character in record
E2F8    20 04 E3    JSR $E304       set pointer to next record
E2FB    90 F4       BCC $E2F1       done in this block?
E2FD    D0 04       BNE $E303       block full?
```

```
E2FF    A9 00       LDA #$00
E301    95 C1       STA $C1,X     write pointer to zero
E303    60          RTS

********************************  set pointer to next record
E304    A6 82       LDX $82       channel number
E306    B5 C1       LDA $C1,X     write pointer
E308    38          SEC
E309    F0 0D       BEQ $E318     equal zero?
E30B    18          CLC
E30C    75 C7       ADC $C7,X     add record length
E30E    90 0B       BCC $E31B     smaller than 256?
E310    D0 06       BNE $E318     equal 256?
E312    A9 02       LDA #$02
E314    2C CC FE    BIT $FECC
E317    60          RTS

E318    69 01       ADC #$01      add two
E31A    38          SEC
E31B    60          RTS

********************************  expand side-sector
E31C    20 D3 D1    JSR $D1D3     get drive number
E31F    20 CB E1    JSR $E1CB     get last side-sector
E322    20 9C E2    JSR $E29C
E325    20 7B CF    JSR $CF7B
E328    A5 D6       LDA $D6
E32A    85 87       STA $87
E32C    A5 D5       LDA $D5       side-sector number
E32E    85 86       STA $86
E330    A9 00       LDA #$00
E332    85 88       STA $88
E334    A9 00       LDA #$00
E336    85 D4       STA $D4
E338    20 0E CE    JSR $CE0E     calculate side-sector no. and ptr
E33B    20 4D EF    JSR $EF4D     number of free blocks
E33E    A4 82       LDY $82       channel number
E340    B6 C7       LDX $C7,Y     record length
E342    CA          DEX
E343    8A          TXA
E344    18          CLC
E345    65 D7       ADC $D7       plus pointer in data block
E347    90 0C       BCC $E355
E349    E6 D6       INC $D6
E34B    E6 D6       INC $D6       increment ptr to end by 2
E34D    D0 06       BNE $E355
E34F    E6 D5       INC $D5       increment side-sector number
E351    A9 10       LDA #$10
E353    85 D6       STA #D6       set pointer to 16
E355    A5 87       LDA $87
E357    18          CLC
E358    69 02       ADC #$02
E35A    20 E9 DE    JSR $DEE9     set buffer ptr for side-sector
E35D    A5 D5       LDA $D5       side-sector number
E35F    C9 06       CMP #$06
```

```
E361   90 05      BCC $E368    smaller than 6?
E363   A9 52      LDA #$52
E365   20 C8 C1   JSR $C1C8    52, 'file too large'
E368   A5 D6      LDA $D6      end pointer
E36A   38         SEC
E36B   E5 87      SBC $87      minus last end pointer
E36D   B0 03      BCS $E372
E36F   E9 0F      SBC #$0F     minus 16
E371   18         CLC
E372   85 72      STA $72
E374   A5 D5      LDA $D5      side-sector number
E376   E5 86      SBC $86      minus last side-sector number
E378   85 73      STA $73      save
E37A   A2 00      LDX #$00
E37C   86 70      STX $70      erase sum for calculation
E37E   86 71      STX $71
E380   AA         TAX
E381   20 51 DF   JSR $DF51    calculate block # of rel-file
E384   A5 71      LDA $71
E386   D0 07      BNE $E38F
E388   A6 70      LDX $70
E38A   CA         DEX
E38B   D0 02      BNE $E38F
E38D   E6 88      INC $88
E38F   CD 73 02   CMP $0273    block number of rel-file
E392   90 09      BCC $E39D    greater than free blocks on disk?
E394   D0 CD      BNE $E363    52, 'file too large'
E396   AD 72 02   LDA $0272
E399   C5 70      CMP $70
E39B   90 C6      BCC $E363    52, 'file too large'
E39D   A9 01      LDA #$01
E39F   20 F6 D4   JSR $D4F6    get byte from buffer
E3A2   18         CLC
E3A3   69 01      ADC #$01     plus 1
E3A5   A6 82      LDX $82
E3A7   95 C1      STA $C1,X    as write pointer
E3A9   20 1E F1   JSR $F11E    find free block in BAM
E3AC   20 FD DD   JSR $DDFD    track and sector in buffer
E3AF   A5 88      LDA $88
E3B1   D0 15      BNE $E3C8    only one block needed?
E3B3   20 5E DE   JSR $DE5E    write block
E3B6   20 1E CF   JSR $CF1E    change buffer
E3B9   20 D0 D6   JSR $D6D0    transmit param to disk controller
E3BC   20 1E F1   JSR $F11E    find free block in BAM
E3BF   20 FD DD   JSR $DDFD    track and sector in buffer
E3C2   20 E2 E2   JSR $E2E2    erase buffer
E3C5   4C D4 E3   JMP $E3D4

E3C8   20 1E CF   JSR $CF1E    change buffer
E3CB   20 D0 D6   JSR $D6D0    transmit param to disk controller
E3CE   20 E2 E2   JSR $E2E2    erase buffer
E3D1   20 19 DE   JSR $DE19    zero byte and end ptr in buffer
E3D4   20 5E DE   JSR $DE5E    write block
E3D7   20 0C DE   JSR $DE0C    get track and sector
E3DA   A5 80      LDA $80      track
```

```
E3DC    48          PHA
E3DD    A4 81       LDA $81        and sector
E3DF    48          PHA            save
E3E0    20 3E DE    JSR $DE3E      get track and sector from disk
E3E3    A5 81       LDA $81        controller
E3E5    48          PHA
E3E6    A5 80       LDA $80        save track and sector
E3E8    48          PHA
E3E9    20 45 DF    JSR $DF45      set buffer ptr for side-sector
E3EC    AA          TAX
E3ED    D0 0A       BNE $E3F9      pointer not zero?
E3EF    20 4E E4    JSR $E44E      write side-sector
E3F2    A9 10       LDA #$10
E3F4    20 E9 DE    JSR $DEE9      buffer pointer to 16
E3F7    E6 86       INC $86        increment side-sector number
E3F9    68          PLA
E3FA    20 8D DD    JSR $DD8D      track in side sector
E3FD    68          PLA
E3FE    20 8D DD    JSR $DD8D      sector in side-sector
E401    68          PLA
E402    85 81       STA $81        sector
E404    68          PLA
E405    85 80       STA $80        and get track back
E407    F0 0F       BEQ $E418      no more blocks?
E409    A5 86       LDA $86        side-sector number
E40B    C5 D5       CMP $D5        changed?
E40D    D0 A7       BNE $E3B6      yes
E40F    20 45 DF    JSR $DF45      set buffer ptr in side-sector
E412    C5 D6       CMP $D6        end pointer
E414    90 A0       BCC $E3B6      smaller?
E416    F0 B0       BEQ $E3C8      same
E418    20 45 DF    JSR $DF45      set buffer ptr in side-sector
E41B    48          PHA
E41C    A9 00       LDA #$00
E41E    20 DC DE    JSR $DEDC      buffer pointer to zero
E421    A9 00       LDA #$00
E423    A8          TAY
E424    91 94       STA ($94),Y    zero as track number
E426    C8          INY
E427    68          PLA            end pointer
E428    38          SEC
E429    E9 01       SBC #$01       minus one
E42B    91 94       STA ($94),Y    as sector
E42D    20 6C DE    JSR $DE6C      write block
E430    20 99 D5    JSR $D599      and verify
E433    20 F4 EE    JSR $EEF4      update BAM
E436    20 0E CE    JSR $CE0E      update pointer for rel-file
E439    20 1E CF    JSR $CF1E      change buffer
E43C    20 F8 DE    JSR $DEF8      right side-sector?
E43F    70 03       BVS $E444      no
E441    4C 75 E2    JMP $E275

E444    A9 80       LDA #$80
E446    20 97 DD    JSR $DD97      set bit 7
E449    A9 50       LDA #$50
```

```
E44B    20 C8 C1    JSR $C1C8       50, 'record not present'

********************************   write side-sector and allocate
                                   new one
E44E    20 1E F1    JSR $F11E       find free block in BAM
E451    20 1E CF    JSR $CF1E       change buffer
E454    20 F1 DD    JSR $DDF1       write block
E457    20 93 DF    JSR $DF93       get buffer number
E45A    48          PHA
E45B    20 C1 DE    JSR $DEC1       erase buffer
E45E    A6 82       LDX $82         channel number
E460    B5 CD       LDA $CD,X       buffer number
E462    A8          TAY
E463    68          PLA
E464    AA          TAX
E465    A9 10       LDA #$10        16 bytes of the side-sector
E467    20 A5 DE    JSR $DEA5       copy in buffer
E46A    A9 00       LDA #$00
E46C    20 DC DE    JSR $DEDC       buffer ptr to 0, old side-sector
E46F    A0 02       LDY #$02
E471    B1 94       LDA ($94),Y     side-sector number
E473    48          PHA
E474    A9 00       LDA #$00
E476    20 C8 D4    JSR $D4C8       buffer ptr to 0, new side-sector
E479    68          PLA
E47A    18          CLC
E47B    69 01       ADC #$01        increment side-sector number
E47D    91 94       STA ($94),Y     and in buffer
E47F    0A          ASL A           times 2
E480    69 04       ADC #$04        plus 4
E482    85 89       STA $89
E484    A8          TAY
E485    38          SEC
E486    E9 02       SBC #$02        minus 2
E488    85 8A       STA $8A         same pointer to old side-sector
E48A    A5 80       LDA $80         track
E48C    85 87       STA $87
E48E    91 94       STA ($94),Y     in buffer
E490    C8          INY
E491    A5 81       LDA $81         sector
E493    85 88       STA $88
E495    91 94       STA ($94),Y     in buffer
E497    A0 00       LDY #$00
E499    98          TYA
E49A    91 94       STA ($94),Y     zero in buffer
E49C    C8          INY
E49D    A9 11       LDA #$11        17
E49F    91 94       STA ($94),Y     number of bytes in block
E4A1    A9 10       LDA #$10        16
E4A3    20 C8 D4    JSR $D4C8       buffer pointer to 16
E4A6    20 50 DE    JSR $DE50       write block
E4A9    20 99 D5    JSR $D599       and verify
E4AC    A6 82       LDX $82         channel number
E4AE    B5 CD       LDA $CD,X       buffer number of the side-sector
E4B0    48          PHA
```

```
E4B1    20 9E DF    JSR $DF9E       get buffer number
E4B4    A6 82       LDX $82         channel number
E4B6    95 CD       STA $CD,X       write in table
E4B8    68          PLA
E4B9    AE 57 02    LDX $0257       channel number + 7
E4BC    95 A7       STA $A7,X       in table
E4BE    A9 00       LDA #$00
E4C0    20 C8 D4    JSR $D4C8       buffer pointer to zero
E4C3    A0 00       LDY #$00
E4C5    A5 80       LDA $80         track
E4C7    91 94       STA ($94),Y     in buffer
E4C9    C8          INY
E4CA    A5 81       LDA $81         sector
E4CC    91 94       STA ($94),Y     in buffer
E4CE    4C DE E4    JMP $E4DE

E4D1    20 93 DF    JSR $DF93       get buffer number
E4D4    A6 82       LDX $82         channel number
E4D6    20 1B DF    JSR $DF1B       read block
E4D9    A9 00       LDA #$00
E4DB    20 C8 D4    JSR $D4C8       buffer pointer to zero
EFDE    C6 8A       DEC $8A
E4E0    C6 8A       DEC $8A         counter for side-sector blocks
E4E2    A4 89       LDY $89
E4E4    A5 87       LDA $87         track number
E4E6    91 94       STA ($94),Y     in buffer
E4E8    C8          INY
E4E9    A5 88       LDA $88         sector number
E4EB    91 94       STA ($94),Y     in buffer
E4ED    20 5E DE    JSR $DE5E       write block
E4F0    20 99 D5    JSR $D599       and verify
E4F3    A4 8A       LDY $8A         counter for side-sector blocks
E4F5    C0 03       CPY #$03
E4F7    B0 D8       BCS $E4D1       greater than or equal to 3?
E4F9    4C 1E CF    JMP $CF1E       change buffer

********************************    table of error messages
E4FC 00                             00
E4FD A0 4F CB                        ' oK'
E500 20 21 22 23 24 27              error numbers of 'read error'
E506 D2 45 41 44                    'Read'
E50A 89                             pointer to 'error'
E50B 52                             52
E50C 83                             pointer to 'file'
E50D 20 54 4F 4F 20 AC 4A 52 47 C5     ' too largE'
E517 50                             50
E518 8B 06                          pointer to 'record ' and 'not '
E51A 20 50 52 45 53 45 4E D4        ' presenT'
E522 51                             51
E523 CF 56 45 52 46 4C 4F 57 20 'Overflow in'
E52E 8B                             pointer to 'record'
E52F 25 28                          error numbers of 'write error'
E531 8A 89                          pointer to 'write' and 'error '
E533 26                             26
E534 8A                             pointer to 'write'
```

```
E535 20 50 52 4F 54 45 43 54 20 4F CE ' protect oN'
E540 29                           29
E541 88                           pointer to 'disk'
E542 20 49 85                     ' id'
E545 85                           pointer to ' mismatch'
E546 30 31 32 33 34               error numbers for 'syntax error'
E54B D3 59 4E 54 41 58            'Syntax'
E551 89                           pointer to ' error'
E552 60                           60
E553 8A 03 84                      ptrs to 'write', 'file' & 'open'
E556 63                           63
E557 83                           pointer to 'file'
E558 20 45 58 49 53 54 D3         ' existS'
E55F 64                           64
E560 83                           pointer to 'file'
E561 20 54 59 50 45               ' type'
E566 85                           pointer to 'mismatch'
E567 65                           65
E568 CE 4F 20 42 4C 4F 43 CB      'No block'
E570 66 67                        'illegal track or sector'
E572 C9 4C 4C 45 47 41 4C 20      'Illegal '
E57A 54 52 41 43 4B 20 4F 52      'track or'
E582 20 53 45 43 54 4F D2         'sectoR'
E589 61                           61
E58A 83 06 84                     pointer to 'file', 'not' & 'open'
E58D 39 62                        error nos. for 'file not found'
E590 83 06 87                      ptrs to 'file', 'not' & 'found'
E593 01                           01
E594 83                           pointer to 'file'
E594 53 20 53 43 52 41 54 43 48 45 C4  's scratcheD'
E59F 70                           70
E5A0 CE 4F 20 43 48 41 4E 4E 45 CC  'No channeL'
E5AA 71                           71
E5AB C4 49 52                     'Dir'
E5AE 89                           pointer to 'error'
E5AF 72                           72
E5B0 88                           pointer to 'disk'
E5B1 20 46 55 4C CC               ' fulL'
E5B6 73                           73
E5B7 C3 42 4D 20 44 4F 53 20      'Cbm dos '
E5BF 56 32 2E 36 20 31 35 34 B1   'v2.6 1541'
E5C4 74                           74
E5C5 C4 42 49 56 45               'Drive'
E5CA 06                           pointer to 'not'
E5CB 20 52 45 41 44 D9            ' readY'
E5D5 09
E5D6 C5 52 52 4F D2               'ErroR'
E5DB 0A
E5DC D7 52 49 54 C5               'WritE'
E5E1 03
E5E2 C6 49 4C C5                  'FilE'
E5E6 04
E6E7 CF 50 45 CE                  'OpeN'
E5EB 05
E5EC CD 49 53 4D 41 54 43 C8      'MismatcH'
```

```
E5F4 06
E5F5 CE 4F D4                        'NoT'
E5F8 07
E5F9 C6 4F 55 4E C4                  'FounD'
E5FE 08
E5FF C4 49 53 CB                     'DisK'
E603 0B
E604 D2 45 43 4F 52 C4               'RecorD'

******************************       prepare error number and message
E60A    48      ,     PHA            save error code
E60B    86 F9         STX $F9        drive number
E60D    8A            TXA
E60E    0A            ASL A          times 2
E60F    AA            TAX            as pointer
E610    B5 06         LDA $06,X
E612    85 80         STA $80        get track
E614    B5 07         LDA $07,X
E616    85 81         STA $81        and sector number
E618    68            PLA            get error code back
E619    29 0F         AND #$0F       isolate bits 0-3
E61B    F0 08         BEQ $E625      zero, then 24, 'read error'
E61D    C9 0F         CMP #$0F       15?
E61F    D0 06         BNE $E627
E621    A9 74         LDA #$74
E623    D0 08         BNE $E62D      74, 'drive not ready'
E625    A9 06         LDA #$06       6
E627    09 20         ORA #$20       add $20
E629    AA            TAX
E62A    CA            DEX
E62B    CA            DEX            subtract two
E62C    8A            TXA
E62D    48            PHA            save error number
E62E    AD 2A 02      LDA $022A      number of the disk command
E631    C9 00         CMP #$00       OPEN or VALIDATE?
E633    D0 0F         BNE $E644      no
E635    A9 FF         LDA #$FF
E637    8D 2A 02      STA $022A
E63A    68            PLA            get error number back
E63B    20 C7 E6      JSR $E6C7      generate error message
E63E    20 42 D0      JSR $D042      load BAM
E641    4C 48 E6      JMP $E648      set error message

E644    68            PLA
E645    20 C7 E6      JSR $E6C7      set error message
E648    20 BD C1      JSR $C1BD      erase input buffer
E64B    A9 00         LDA #$00
E64D    8D F9 02      STA $02F9      erase error flag
E650    20 2C C1      JSR $C12C      turn LED off
E653    20 DA D4      JSR $D4DA      close channels 17 and 18
E656    A9 00         LDA #$00
E658    85 A3         STA $A3        input buffer pointer to zero
E65A    A2 45         LDX #$45
E65C    9A            TXS            initialize stack pointer
E65D    A5 84         LDA $84        secondary address
```

```
E65F    29 0F       AND #$0F
E661    85 83       STA $83
E663    C9 0F       CMP #$0F       15?
E665    F0 31       BEQ $E698      yes, command channel
E667    78          SEI
E668    A5 79       LDA $79        LISTEN active?
E66A    D0 1C       BNE $E688      yes
E66C    A5 7A       LDA $7A        TALK active?
E66E    D0 10       BNE $E680      yes
E670    A6 83       LDX $83        channel number
E672    BD 2B 02    LDA $022B,X    open channel to this second. addr
E675    C9 FF       CMP #$FF
E677    F0 1F       BEQ $E698      no
E679    29 0F       AND #$0F
E67B    85 82       STA $82        channel number
E67D    4C 8E E6    JMP $E68E

********************************   TALK
E680    20 EB D0    JSR $D0EB      open channel for reading
E683    20 4E EA    JSR $EA4E      accept byte
E686    D0 06       BNE $E68E

********************************   LISTEN
E688    20 07 D1    JSR $D107      open channel for writing
E68B    20 4E EA    JSR $EA4E      accept byte
E68E    20 25 D1    JSR $D125      verify file type
E691    C9 04       CMP #$04       file type REL?
E693    B0 03       BCS $E698      yes
E695    20 27 D2    JSR $D227      close channel
E698    4C E7 EB    JMP $EBE7

********************************   convert hex to decimal (2 bytes)
E69B    AA          TAX
E69C    A9 00       LDA #$00
E69E    F8          SED
E69F    E0 00       CPX #$00
F6A1    F0 07       BEQ $E6AA      convert hex to BCD
E6A3    18          CLC
E6A4    69 01       ADC #$01
A6A6    CA          DEX
E6A7    4C 9F E6    JMP $E69F
E6AA    D8          CLD

********************************   divide BCD number into two bytes
E6AB    AA          TAX
E6AC    4A          LSR A
E6AD    4A          LSR A          shift hi-nibble down
E6AE    4A          LSR A
E6AF    4A          LSR A
E6B0    20 B4 E6    JSR $E6B4      convert to ASCII
E6B3    8A          TXA
E6B4    29 0F       AND #$0F       erase top 4 bits
E6B6    09 30       ORA #$30       add '0'
E6B8    91 A5       STA ($A5),Y    write in buffer
E6BA    C8          INY            increment buffer pointer
```

```
E6BB    60          RTS

******************************      write 'ok' in buffer
E6BC    20 23 C1    JSR $C123       erase error flag
E6BF    A9 00       LDA #$00        error number 0
E6C1    A0 00       LDY #$00
E6C3    84 80       STY $80         track 0
E6C5    84 81       STY $81         sector 0

******************************      error message in buffer
E6C7    A0 00       LDY #$00        buffer pointer
E6C9    A2 D5       LDX #$D5
E6C8    86 A5       STX $A5         pointer $A5/$A6 TO $2D5
E6CD    A2 02       LDX #$02
E6CF    86 A6       STX $A6
E6D1    20 AB E6    JSR $E6AB       error # to ASCII and in buffer
E6D4    A9 2C       LDA #$2C        ',' comma
E6D6    9A A5       STA ($A5),Y     write in buffer
ED68    C8          INY             increment buffer pointer
E6D9    AD D5 02    LDA $02D5       first digit of the disk status
E6DC    8D 43 02    STA $0243       in output register
E6DF    8A          TXA             error number in accumulator
E6E0    20 06 E7    JSR $E706       error message in buffer
E6E3    A9 2C       LDA #$2C        ',' comma
E6E5    91 A5       STA ($A5),Y     write in buffer
E6E7    C8          INY             and increment buffer pointer
E6E8    A5 80       LDA $80         track number
E6EA    20 9B E6    JSR $E69B       to ASCII and in buffer
E6ED    A9 2C       LDA #$2C        ',' comma
E6EF    91 A5       STA ($A5),Y     write in buffer
E6F1    C8          INY             increment buffer pointer
E6F2    A5 81       LDA $81         sector
E6F4    20 9B E6    JSR $E69B       convert to ASCII and in buffer
E6F7    88          DEY
E6F8    98          TYA
E6F9    18          CLC
E6FA    69 D5       ADC #$D5
E6FC    8D 49 02    STA $0249       end pointer
E6FF    E6 A5       INC $A5
E701    A9 88       LDA #$88        set READ flag
E703    85 F7       STA $F7
E705    60          RTS

******************************      write error message to buffer
E706    AA          TAX             error code to X
E707    A5 86       LDA $86
E709    48          PHA             preserve pointer $86/$87
E70A    A5 87       LDA $87
E70C    48          PHA
E70D    A9 FC       LDA #$FC
E70F    85 86       STA #$E4        start of the error messages
E713    85 87       STA $87
E715    8A          TXA             error number in accumulator
E716    A2 00       LDX #$00
E718    C1 86       CMP ($86,X)     compare with error no in table
```

```
E71A   F0 21      BEQ $E73D
E71C   48         PHA
E71D   20 75 E7   JSR $E775      bit 7 into carry and erase
E720   90 05      BCC $E727      not set?
E722   20 75 E7   JSR $E775      bit 7 into carry
E725   90 FB      BCC $E722      wait for character with bit 7 set
E727   A5 87      LDA $87
E729   C9 E6      CMP #$E6
E72B   90 08      BCC $E735      $E60A, check to end of table
E72D   D0 0A      BNE $E739
E72F   A0 0A      LDA #$0A
E731   C5 86      CMP $86
E733   90 04      BCC $E739
E735   68         PLA
E736   4C 18 E7   JMP $E718      no, continue
E739   68         PLA
E73A   4C 4D E7   JMP $E74D      done

E73D   20 67 E7   JSR $E767      get a character, bit 7 in carry
E740   90 FB      BCC $E73D      wait for character with bit 7 set
E742   20 54 E7   JSR $E754      and write in buffer
E745   20 67 E7   JST $E767      get next character
E748   90 F8      BCC $E742      wait for character with bit 7 set
E74A   20 54 E7   JSR $E754      put character in buffer
E74D   68         PLA
E74E   85 87      STA $87
E750   68         PLA            get pointer $86/$87 back
E751   85 86      STA $86
E753   60         RTS

********************************  get character and in buffer
E754   C9 20      CMP #$20       ' ' blank
E756   B0 0B      BCS $E763      greater, then write in buffer
E758   AA         TAX            save code
E759   A9 20      LDA #$20       blank
E75B   91 A5      STA ($A5),Y    write in buffer
E75D   C8         INY            increment buffer pointer
E75E   8A         TXA            code in accumulator
E75F   20 06 E7   JSR $E706      output previous text
E762   60         RTS
E763   91 A5      STA ($A5),Y    write character in buffer
E765   C8         INY            and increment pointer
E766   60         RTS

********************************  get a char of the error message
E767   E6 86      INC $86
E769   D0 02      BNE $E76D      increment pointer
E76B   E6 87      INC $87
E76D   A1 86      LDA ($86,X)    get character
E76F   0A         ASL A          bit 7 into carry
E770   A1 86      LDA ($86,X)    get character
E772   29 7F      AND #$7F       erase bit 7
E774   60         PTS

********************************  increment pointer
```

```
E775    20 6D E7    JSR $E76D      bit 7 into carry
E778    E6 86       INC $86
E77A    D0 02       BNE $E77E      increment pointer
E77C    E6 87       INC $87
E77E    60          RTS

********************************
E77F    60          RTS

********************************   check for AUTO-start
E780    AD 00 18    LDA $1800      read IEEE port
E783    AA          TAX
E784    29 04       AND #$04       isolate 'CLOCK IN' bit
E786    F0 F7       BEQ $E77F      not set, then done
E788    8A          TXA
E789    29 01       AND #$01       isolate 'DATA IN' bit
E78B    F0 F2       BEQ $E77F      not set, then done
E78D    58          CLI
E78E    AD 00 18    LDA $1800      load IEEE port
E791    29 05       AND #$05       test 'DATA IN' and 'CLOCK IN'
E793    F0 F9       BNE $E78E      wait until both set
E795    EE 78 02    INC $0278      file name
E798    EE 74 02    INC $0274      character in the input line
E79B    A9 2A       LDA #$2A       '*' as filename
E79D    8D 00 02    STA $0200      write in buffer
E7A0    4C A8 E7    JMP $E7A8

********************************   '&' - command
E7A3    A9 8D       LDA #$8D
E7A5    20 68 C2    JSR $C268      check command line to end
E7A8    20 58 F2    JSR $F258      (RTS)
E7AB    AD 78 02    LDA $0278      number of file names
E7AE    48          PHA            save
E7AF    A9 01       LDA #$01
E7B1    8D 78 02    STA $0278      file name
E7B4    A9 FF       LDA #$FF
E7B6    85 86       STA $86
E7B8    20 4F C4    JSR $C44F      find file
E7BB    AD 80 02    LDA $0280
E7BE    D0 05       BNE $E7C5      found?
E7C0    A9 39       LDA #$39
E7C2    20 C8 C1    JSR $C1C8      39, 'file not found'
E7C5    68          PLA
E7C6    8D 78 02    STA $0278      get number of file names back
E7C9    AD 80 02    LDA $0280
E7CC    85 80       STA $80        track
E7CE    AD 85 02    LDA $0285
E7D1    85 81       STA $81        and sector
E7D3    A9 03       LDA #$03       file type 'USR'
E7D5    20 77 D4    JSR $D477      buffer allocated, read 1st block
E7D8    A9 00       LDA #$00
E7DA    85 87       STA $87        erase checksum
E7DC    20 39 E8    JSR $E839      get byte from file
E7DF    85 88       STA $88        save as start address lo
E7E1    20 4B E8    JSR $E84B      form checksum
```

```
E7E4    20 39 E8    JSR $E839      get byte from file
E7E7    85 89       STA $89        as start address hi
E7E9    20 4B E8    JSR $E84B      form checksum
E7EC    A5 86       LDA $86
E7EE    F0 0A       BEQ $E7FA
E7F0    A5 88       LDA $88
E7F2    48          PHA            save program start address
E7F3    A5 89       LDA $89
E7F5    48          PHA
E7F6    A9 00       LDA #$00
E7F8    85 86       STA $86
E7FA    20 39 E8    JSR $E839      get byte from file
E7FD    85 8A       STA $8A        save as counter
E7FF    20 4B E8    JSR $E84B      form checksum
E802    20 39 E8    JSR $E839      get byte from file
E805    A0 00       LDY #$00
E807    91 88       STA ($88),Y    save as program bytes
E809    20 4B E8    JSR $E84B      form checksum
E80C    A5 88       LDA $88
E80E    18          CLC
E80F    69 01       ADC #$01
E811    85 88       STA $88        increment $88/$89
E813    90 02       BCC $E817
E815    E6 89       INC $89
E817    C6 8A       DEC $8A        decrement pointer
E819    D0 E7       BNE $E802
E81B    20 35 CA    JSR $CA35      get next byte
E81E    A5 85       LDA $85        data byte
E820    C5 87       CMP $87        equal to checksum?
E822    F0 08       BEQ $E82C      yes
E824    20 3E DE    JSR $DE3E      transmit param to disk controller
E827    A9 50       LDA #$50
E829    20 45 E6    JSR $E645      50, 'record not present'
E82C    A5 F8       LDA $F8        end?
E82E    D0 A8       BNE $E7D8      no, next data block
E830    68          PLA
E831    85 89       STA $89
E833    68          PLA            get program start address back
E834    85 88       STA $88
E836    6C 88 00    JMP ($0088)    and execute program
E839    20 35 CA    JSR $CA35      get byte from file
E83C    A5 F8       LDA $F8        end?
E83E    D0 08       BNE $E848      no
E840    20 3E DE    JSR $DE3E      transmit param to disk controller
E843    A9 51       LDA #$51
E845    20 45 E6    JSR $E645      51, 'overflow in record'
E848    A5 85       LDA $85        data byte
E84A    60          RTS

********************************   generate checksum
E84B    A8          CLC
E84C    65 87       ADC $87
E84E    69 00       ADC #$00
E850    85 87       STA $87
E852    60          RTS
```

```
********************************     IRQ routine for serial bus
E853   AD 01 18    LDA $1801         read port A, erase IRQ flag
E856   A9 01       LDA #$01
E858   85 7C       STA $7C           set flag for 'ATN received'
E85A   60          RTS

********************************     servicing the serial bus
E85B   78          SEI
E85C   A9 00       LDA #$00
E85E   85 7C       STA $7C           erase flag for 'ATN received'
E860   85 79       STA $79           erase flag for LISTEN
E862   85 7A       STA $7A           erase flag for TALK
E864   A2 45       LDX #$45
E866   9A          TXS               initialize stack pointer.
E867   A9 80       LDA #$80
E869   85 F8       STA $F8           erase end flag
E86B   85 7D       STA $7D           erase EOI flag
E86D   20 B7 E9    JSR $E9B7         CLOCK OUT lo
E870   20 A5 E9    JSR $E9A5         DATA OUT, bit '0', hi
E873   AD 00 18    LDA $1800
E876   09 10       ORA #$10          switch data lines to input
E878   8D 00 18    STA $1800
E87B   AD 00 18    LDA $1800         read IEEE port
E87E   10 57       BPL $E8D7         EOI?
E880   29 04       AND #$04          CLOCK IN?
E882   D0 F7       BNE $E87B         no
E884   20 C9 E9    JSR $E9C9         get byte from bus
E887   C9 3F       CMP #$3F          unlisten?
E889   D0 06       BNE $E891         no
E88B   A9 00       LDA #$00
E88D   85 79       STA $79           reset flag for LISTEN
E88F   F0 71       BEQ $E902
E891   C9 5F       CMP #$5F          untalk?
E893   D0 06       BNE $E89B         no
E895   A9 00       LDA #$00
E897   85 7A       STA $7A           reset flag for TALK
E899   F0 67       BEQ $E902
E89B   C5 78       CMP $78           TALK address?
E89D   D0 0A       BNE $E8A9         no
E89F   A9 01       LDA #$01
E8A1   85 7A       STA $7A           set flag for TALK
E8A3   A9 00       LDA #$00
E8A5   85 79       STA $79           reset flag for LISTEN
E8A7   F0 29       BEQ $E8D2
E8A9   C5 77       CMP $77           LISTEN address?
E8AB   D0 0A       BNE $E8B7         no
E8AD   A9 01       LDA #$01
E8AF   85 79       STA $79           set flag for LISTEN
E8B1   A9 00       LDA #$00
E8B3   85 7A       STA $7A           reset flag for TALK
E8B5   F0 1B       BEQ $E8D2
E8B7   AA          TAX
E8B8   29 60       AND #$60
E8BA   C9 60       CMP #$60          set bit 5 and 6
```

```
E8BC    D0 3F       BNE $E8FD      no
E8BE    8A          TXA
E8BF    85 84       STA $84        byte is secondary address
E8C1    29 0F       AND #$0F
E8C3    85 83       STA $83        channel number
E8C5    A5 84       LDA $84
E8C7    29 F0       AND #$F0
E8C9    C9 E0       CMP #$E0       CLOSE?
E8CB    D0 35       BNE $E902
E8CD    58          CLI
E8CE    20 C0 DA    JSR $DAC0      CLOSE routine
E8D1    78          SEI
E8D2    2C 00 18    BIT $1800
E8D5    30 AD       BMI $E884
E8D7    A9 00       LDA #$00
E8D9    85 7D       STA $7D        set EOI
E8DB    AD 00 18    LDA $1800      IEEE port
E8DE    29 EF       AND #$EF       switch data lines to output
E8E0    8D 00 18    STA $1800
E8E3    A5 79       LDA $79        LISTEN active?
E8E5    F0 06       BEQ $E8ED      no
E8E7    20 2E EA    JSR $EA2E      receive data
E8EA    4C E7 EB    JMP $EBE7      to delay loop

E8ED    A5 7A       LDA $7A        TALK active?
E8EF    F0 09       BEQ $E8FA      no
E8F1    20 9C E9    JSR $E99C      DATA OUT, bit '1', lo
E8F4    20 AE E9    JSR $E9AE      CLOCK OUT hi
E8F7    20 09 E9    JSR $E909      send data
E8FA    4C 4E EA    JMP $EA4E      to delay loop
E8FD    A9 10       LDA #$10       either TALK or LISTEN,ignore byte
E8FF    8D 00 18    STA $1800      switch data lines to input
E902    2C 00 18    BIT $1800
E905    10 D0       BPL $E8D7
E907    30 F9       BMI $E902      wait for handshake

********************************   send data
E909    78          SEI
E90A    20 EB D0    JSR $D0EB      open channel for read
E90D    B0 06       BCS $E915      channel active
E90F    A6 82       LDX $82        channel number
E911    B5 F2       LDA $F2,X      set READ flag?
E913    30 01       BMI $E916      yes
E915    60          RTS
E916    20 59 EA    JSR $EA59      check EOI
E919    20 C0 E9    JSR $E9C0      read IEEE port
E91C    29 01       AND #$01       isolate data bit
E91E    08          PHP            and save
E91F    20 B7 E9    JSR $E9B7      CLOCK OUT lo
E922    28          PLP
E923    F0 12       BEQ $E937
E925    20 59 EA    JSR $EA59      check EOI
E928    20 C0 E9    JSR $E9C0      read IEEE port
E92B    29 01       AND #$01       isolate data bit
E92D    D0 F6       BNE $E925
```

```
E92F    A6 82       LDX $82        channel number
E931    B5 F2       LDA $F2,X
E933    29 08       AND #$08
E935    D0 14       BNE $E94B
E937    20 59 EA    JSR $EA59      check EOI
E93A    20 C0 E9    JSR $E9C0      read IEEE port
E93D    29 01       AND #$01       isolate data bit
E93F    D0 F6       BNE $E937
E941    20 59 EA    JSR $EA59      check EOI
E944    20 C0 E9    JSR $E9C0      read IEEE port
E947    29 01       AND #$01       isolate data bit
E949    F0 F6       BEQ $E941
E84B    20 AE E9    JSR $E9AE      CLOCK OUT hi
E94E    20 59 EA    JSR $EA59      check EOI
E951    20 C0 E9    JSR $E9C0      read IEEE port
E954    29 01       AND #$01       isolate data bit
E956    D0 F3       BNE $E94B
E958    A9 08       LDA #$08       counter to 8 bits for serial
E95A    85 98       STA $98        transmission
E95C    20 C0 E9    JSR $E9C0      read IEEE port
E95F    29 01       AND #$01       isolate data bit
E961    D0 36       BNE $E999
E963    A6 82       LDX $82
E965    BD 3E 02    LDA $023E,X
E968    6A          ROR A          lowest bit in carry
E969    9D 3E 02    STA $023E,X
E96C    B0 05       BCS $E973      set bit
E96E    20 A5 E9    JSR $E9A5      DATA OUT, output bit '0'
E971    D0 03       BNE $E976      absolute jump
E973    20 9C E9    JSR $E99C      DATA OUT, output bit '1'
E976    20 B7 E9    JSR $E9B7      set CLOCK OUT
E979    A5 23       LDA $23
E97B    D0 03       BNE $E980
E97D    20 F3 FE    JSR $FEF3      delay for serial bus
E980    20 FB FE    JSR $FEFB      set DATA OUT and CLOCK OUT
E983    C6 98       DEC $98        all bits output?
E985    D0 D5       BNE $E95C      no
E987    20 59 EA    JSR $EA59      check EOI
E98A    20 C0 E9    JSR $E9C0      read IEEE port
E98D    29 01       AND #$01       isolate data bit
E98F    F0 F6       BEQ $E987
E991    58          CLI
E992    20 AA D3    JSR $D3AA      get next data byte
E995    78          SEI
E996    4C 0F E9    JMP $E90F      and output

E999    4C 4E EA    JMP $EA4E      to delay loop

********************************   DATA OUT lo
E99C    AD 00 18    LDA $1800
E99F    29 FD       AND #$FD       output bit '1'
E9A1    8D 00 18    STA $1800
E9A4    60          RTS

********************************   DATA OUT hi
```

```
E9A5    AD 00 18    LDA $1800
E9A8    09 02       ORA #$02       output bit '0'
E9AA    8D 00 18    STA $1800
E9AD    60          RTS

********************************   CLOCK OUT hi
E9AE    AD 00 18    LDA $1800
E9B1    09 08       ORA #$08       set bit 3
E9B3    8D 00 18    STA $1800
E9B6    60          RTS

********************************   CLOCK OUT lo
E9B7    AD 00 18    LDA $1800
E9BA    29 F7       AND #$F7       erase bit 3
E9BC    8D 00 18    STA $1800
E9BF    60          RTS

********************************   read IEEE port
E9C0    AD 00 18    LDA $1800      read port
E9C3    CD 00 18    CMP $1800      wait for constants
E9C6    D0 F8       BNE $E9C0
E9C8    60          RTS

********************************
E9C9    A9 08       LDA #$08
E9CB    85 98       STA $98        bit counter for serial output
E9CD    20 59 EA    JSR $EA59      check EOI
E9D0    20 C0 E9    JSR $E9C0      read IEEE port
E9D3    29 04       AND #$04       CLOCK IN?
E9D5    D0 F6       BNE $E9CD      no, wait
E9D7    20 9C E9    JSR $E99C      DATA OUT, bit '1'
E9DA    A9 01       LDA #$01
E9DC    8D 05 18    STA $1805      set timer
E9DF    20 59 EA    JSR $EA59      check EOI
E9E2    AD 0D 18    LDA $180D
E9E5    29 40       AND #$40       timer run down?
E9E7    D0 09       BNE $E9F2      yes, EOI
E9E9    20 C0 E9    JSR $E9C0      read IEEE port
E9EC    29 04       AND #$04       CLOCK IN?
E9EE    F0 EF       BEQ $E9DF      no, wait
E9F0    D0 19       BNE $EA0B
E9F2    20 A5 E9    JSR $E9A5      DATA OUT bit '0' hi
E9F5    A2 0A       LDY #$0A       10
E9F7    CA          DEX            delay loop, approx 50 micro sec.
E9F8    D0 FD       BNE $E9F7
E9FA    20 9C E9    JSR $E99C      DATA OUT, bit '1', lo
E9FD    20 59 EA    JSR $EA59      check EOI
EA00    20 C0 E9    JSR $E9C0      read IEEE
EA03    29 04       AND #$04       CLOCK IN?
EA05    F0 F6       BEQ $E9FD      no, wait
EA07    A9 00       LDA #$00
EA09    85 F8       STA $F8        set EOI flag
EA0B    AD 00 18    LDA $1800      IEEE port
EA0E    49 01       EOR #$01       invert data byte
EA10    4A          LSR A
```

```
EA11    29 02       AND #$02
EA13    D0 F6       BNE $EA0B       CLOCK IN?
EA15    EA          NOP
EA16    EA          NOP
EA17    EA          NOP
EA18    66 85       ROR $85         prepare next bit
EA1A    20 59 EA    JSR $EA59       check EOI
EA1D    20 C0 E9    JSR $E9C0       read IEEE port
EA20    29 04       AND #$04        CLOCK IN?
EA22    F0 F6       BEQ $EA1A       no
EA24    C6 98       DEC $98         decrement bit counter
EA26    D0 E3       BNE $EA0B       all bits output?
EA28    20 A5 E9    JSR $E9A5       DATA OUT, bit '0', hi
EA2B    A5 85       LDA $85         load data byte again
EA2D    60          RTS

********************************    accept data from serial bus
EA2E    78          SEI
EA2F    20 07 D1    JSR $D107       open channel for writing
EA32    B0 05       BCS $EA39       channel not active?
EA34    B5 F2       LDA $F2,X       WRITE flag
EA36    6A          ROR A
EA37    B0 0B       BCS $EA44       not set?
EA39    A5 84       LDA $84         secondary address
EA3B    29 F0       AND #$F0
EA3D    C9 F0       CMP #$F0        OPEN command?
EA3F    F0 03       BEQ $EA44       yes
EA41    4C 4E EA    JMP $EA4E       to wait loop

EA44    20 C9 E9    JSR $E9C9       get data byte from bus
EA47    58          CLI
EA48    20 B7 CF    JSR $CFB7       and write in buffer
EA4B    4C 2E EA    JMP $EA2E       to loop beginning

EA4E    A9 00       LDA #$00
EA50    8D 00 18    STA $1800       reset IEEE port
EA53    4C E7 EB    JMP $EBE7       to wait loop

EA56    4C 5B E8    JMP $EB58       to serial bus main loop

********************************
EA59    A5 7D       LDA $7D         EOI received?
EA5B    F0 06       BEQ $EA63       yes
EA5D    AD 00 18    LDA $1800       IEEE port
EA60    10 09       BPL $EA6B

EA62    60          RTS

EA63    AD 00 18    LDA $1800       IEEE port
EA66    10 FA       BPL $EA62
EA68    4C D7 E8    JMP $E8D7       set EOI, serve serial bus

********************************    blink LED for hardware defects
EA6E    A2 00       LDX #$00        blink once, zero page
EA70    2C          .BYTE $2C
```

```
EA71   A5 6F      LDX $6F      blink X+1 times for RAM/ROM err
EA73   9A         TXS
EA74   BA         TSX
EA75   A9 08      LDA #$08     select LED bit in the port
EA77   0D 00 1C   ORA $1C00
EA7A   4C EA FE   JMP $FEEA    turn LED on, back to $EA7D
EA7D   98         TYA
EA7E   18         CLC
EA7F   69 01      ADC #$01
EA81   D0 FC      BNE $EA7F
EA83   88         DEY
EA84   D0 F8      BNE $EA7E
EA86   AD 00 1C   LDA $1C00
EA89   29 F7      AND #$F7     turn LED off
EA8B   8D 00 1C   STA $1C00
EA8E   98         TYA
EA8F   18         CLC
EA90   69 01      ADC #$01
EA92   D0 FC      BNE $EA90    delay loop
EA94   88         DEY
EA95   D0 F8      BNE $EA8F
EA97   CA         DEX
EA98   10 DB      BPL $EA75
EA9A   E0 FC      CPX #$FC
EA9C   D0 F0      BNE $EA8E    wait for delay
EA9E   F0 D4      BEQ $EA74    turn LED on again

********************************   RESET routine
EAA0   78         SEI
EAA1   D8         CLD
EAA2   A2 FF      LDX #$FF
EAA4   8E 03 18   STX $1803    port A to output
EAA7   E8         INX
EAA8   A0 00      LDY #$00
EAAA   A2 00      LDX #$00
EAAC   8A         TXA
EAAD   95 00      STA $00,X    erase zero page
EAAF   E8         INX
EAB0   D0 FA      BNE $EAAC
EAB2   8A         TXA
EAB3   D5 00      CMP $00,X    is byte erased?
EAB5   D0 B7      BNE $EA6E    no, then to error display (blink)
EAB7   F6 00      INC $00,X
EAB9   C8         INY
EABA   D0 FB      BNE $EAB7
EABC   D5 00      CMP $00,X
EABE   D0 AE      BNE $EA6E    error
EAC0   94 00      STY $00,X
EAC2   B5 00      LDA $00,X
EAC4   D0 A8      BNE $EA6E    error
EAC6   E8         INX
EAC7   D0 E9      BNE $EAB2
EAC9   E6 6F      INC $6F
EACB   86 76      STX $76
EACD   A9 00      LDA #$00
```

```
EACF    85 75       STA $75
EAD1    A8          TAY
EAD2    A2 20       LDX #$20       test 32 pages
EAD4    18          CLC
EAD5    C6 76       DEC $76
EAD7    71 75       ADC ($75),Y
EAD9    C8          INY
EADA    D0 FB       BNE $EAD7
EADC    CA          DEX
EADD    D0 F6       BNE $EAD5      test ROM
EADF    69 00       ADC #$00
EAE1    AA          TAX
EAE2    C5 76       CMP $76
EAE4    D0 39       BNE $EB1F      ROM error
EAE6    E0 C0       CPX #$C0
EAE8    D0 DF       BNE $EAC9
EAEA    A9 01       LDA #$01
EAEC    85 76       STA $76
EAEE    E6 6F       INC $6F
EAF0    A2 07       LDX #$07       test RAM, beginning at page 7
EAF2    98          TYA
EAF3    18          CLC
EAF4    65 76       ADC $76
EAF6    91 75       STA ($75),Y
EAF8    C8          INY
EAF9    D0 F7       BNE $EAF2
EAFB    E6 76       INC $76
EAFD    CA          DEX
EAFE    D0 F2       BNE $EAF2
EB00    A2 07       LDX #$07
EB02    C6 76       DEC $76
EB04    88          DEY
EB05    98          TYA
EB06    18          CLC
EB07    65 76       ADC $76
EB09    D1 75       CMP ($75),Y
EB0B    D0 12       BNE $EB1F      RAM error
EB0D    49 FF       EOR #$FF
EB0F    91 75       STA ($75),Y
EB11    51 75       EOR ($75),Y
EB13    91 75       STA ($75),Y
EB15    D0 08       BNE $EB1F      RAM error
EB17    98          TYA
EB18    D0 EA       BNE $EB04
EB1A    CA          DEX
EB1B    D0 E5       BNE $EB02      continue test
EB1D    F0 03       BEQ $EB22      ok
EB1F    4C 71 EA    JMP $EA71      to error display

EB22    A2 45       LDX #$45
EB24    9A          TXS            initialize stack pointer
EB25    AD 00 1C    LDA $1C00
EB28    29 F7       AND #$F7       turn LED off
EB2A    8D 00 1C    STA $1C00
EB2D    A9 01       LDA #$01
```

```
EB2F    8D 0C 18    STA $180C      CA1 (ATN IN) trigger on pos edge
EB32    A9 82       LDA #$82
EB34    8D 0D 18    STA $180D      interrupt possible through ATN IN
EB37    8D 0E 18    STA $180E
EB3A    AD 00 18    LDA $1800      read port B
EB3D    29 60       AND #$60       isolate bits 5 & 6 (device #)
EB3F    0A          ASL A
EB40    2A          ROL A
EB41    2A          ROL A          rotate to bit positions 0 & 1
EB42    2A          ROL A
EB43    09 48       ORA #$48        add offset from 8 + $40 for TALK
EB45    85 78       STA $78        device number for TALK (send)
EB47    49 60       EOR #$60       erase bit 6, set bit 5
EB49    85 77       STA $77        device number + $20 for LISTEN
EB4B    A2 00       LDX #$00
EB4D    A0 00       LDY #$00
EB4F    A9 00       LDA #$00
EB51    95 99       STA $99,X      low-byte of buffer address
EB53    E8          INX
EB54    B9 E0 FE    LDA $FEE0,Y    high byte of address from table
EB57    95 99       STA $99,X      save
EB59    E8          INX
EB5A    C8          INY
EB5B    C0 05       CPY #$05
EB5D    D0 F0       BNE $EB4F
EB5F    A9 00       LDA #$00
EB61    95 99       STA $99,X
EB63    E8          INX            ptr $A3/$A4 to $200, input buffer
EB64    A9 02       LDA #$02
EB66    95 99       STA $99,X
EB68    E8          INX
EB69    A9 D5       LDA #$D5
E86B    95 99       STA $99,X
EB6D    E8          INX            pointer $A5/$A6 to $2D5, error
EB6E    A9 02       LDA #$02       message pointer
EB70    95 99       STA $99,X
EB72    A9 FF       LDA #$FF
EB74    A2 12       LDX #$12
EB76    9D 2B 02    STA $022B,X     fill channel table with $FF
EB79    CA          DEX
EB7A    10 FA       BPL $EB76
EB7C    A2 05       LDX #$05
EB7E    95 A7       STA $A7,X      erase buffer table
EB80    95 AE       STA $AE,X
EB82    95 CD       STA $CD,X      erase side-sector table
EB84    CA          DEX
EB85    10 F7       BPL $EB7E
EB87    A9 05       LDA #$05       buffer 5
EB89    85 AB       STA $AB        associate with channel 4
EB8B    A9 06       LDA #$06       buffer 6
EB8D    85 AC       STA $AC        associate with channel 5
EB8F    A9 FF       LDA #$FF
EB91    85 AD       STA $AD
EB93    85 B4       STA $B4
EB95    A9 05       LDA #$05
```

```
EB97   8D 3B 02   STA $023B     channel 5 WRITE flag erased
EB9A   A9 84      LDA #$84
EB9C   8D 3A 02   STA $023A     channel 4 WRITE flag set
EB9F   A9 0F      LDA #$0F      initialize channel allocation reg
EBA1   8D 56 02   STA $0256     bit '1' equals channel free
EBA4   A9 01      LDA #$01
EBA6   85 F6      STA $F6       WRITE flag
EBA8   A9 88      LDA #$88
EBAA   85 F7      STA $F7       READ flag
EBAC   A9 F0      LDA #$E0      5 buffers free
EBAE   8D 4F 02   STA $024F     initialize buffer allocation reg
EBB1   A9 FF      LDA #$FF      $24F/$250, 16 bit
EBB3   8D 50 02   STA $0250
EBB6   A9 01      LDA #$01
EBB8   85 1C      STA $1C       flags for WRITE protect
EBBA   85 1D      STA $1D
EBBC   20 63 CB   JSR $CB63     set vector for U0
EBBF   20 FA CE   JSR $CEFA     initialize channel table
EBC2   20 59 F2   JSR $F259     intialization for disk controller
EBC5   A9 22      LDA #$22
EBC7   85 65      STA $65
EBC9   A9 EB      LDA #$EB      pointer $65/$66 to $EB22
EBCB   85 66      STA $66
EBCD   A9 0A      LDA #$0A
EBCF   85 69      STA $69       step width 10
EBD1   A9 05      LDA #$05      for sector assignment
EBD3   85 6A      STA $6A       5 read attempts
EBD5   A9 73      LDA #$73      prepare power-up message
EBD7   20 C1 E6   JSR $E6C1     73, 'cbm dos v2.6 1541'
EBDA   A9 1A      LDA #$1A      bit 1, 3 & 4 to exit
EBDC   8D 02 18   STA $1802     data direction of port B
EBDF   A9 00      LDA #$00
EBE1   8D 00 18   STA $1800     erase data register
EBE4   20 80 E7   JSR $E780     check for auto-start
EBE7   58         CLI
EBE8   AD 00 18   LDA $1800
EBEB   29 E5      AND #$E5      reset serial port
EBED   8D 00 18   STA $1800
EBF0   AD 55 02   LDA $0255     command flag set?
EBF3   F0 0A      BEQ $EBFF     no
EBF5   A9 00      LDA #$00
EBF7   8D 55 02   STA $0255     reset command flag
EBFA   85 67      STA $67
EBFC   20 46 C1   JSR $C146     analyze and execute command

********************************  wait loop
EBFF   58         CLI
EC00   A5 7C      LDA $7C       ATN signal discovered?
EC02   F0 03      BEQ $EC07     no
EC04   4C 5B E8   JMP $E85B     to IEEE routine
EC07   58         CLI
EC08   A9 0E      LDA #$0E      14
EC0A   85 72      STA $72       as secondary address
EC0C   A9 00      LDA #$00
EC0E   85 6F      STA $6F       job counter
```

```
EC10   85 70      STA $70
EC12   A6 72      LDX $72
EC14   BD 2B 02   LDA $022B,X    secondary address
EC17   C9 FF      CMP #$FF       channel associated?
EC19   F0 10      BEQ $EC2B      no
EC1B   26 3F      AND #$3F
EC1D   85 82      STA $82        channel number
EC1F   20 93 DF   JSR $DF93      get buffer number
EC22   AA         TAX
EC23   BD 5B 02   LDA $025B,X    drive number
EC26   29 01      AND #$01
EC28   AA         TAX
EC29   F6 6F      INC $6F,X      increment job counter
EC2B   C6 72      DEC $72        lo address
EC2D   10 E3      BPL $EC12      continue search
EC2F   A0 04      LDY #$04       buffer counter
EC31   B9 00 00   LDA $0000,Y    disk controller in action?
EC34   10 05      BPL $EC3B      no
EC36   29 01      AND #$01       isolate drive number
EC38   AA         TAX
EC39   F6 6F      INC $6F,X      increment job counter
EC3B   88         DEY
EC3C   10 F3      BPL $EC31      next buffer
EC3E   78         SEI
EC3F   AD 00 1C   LDA $1C00
EC42   29 F7      AND #$F7       erase LED bit
EC44   48         PHA
EC45   A5 7F      LDA $7F        drive number
EC47   85 86      STA $86
EC49   A9 00      LDA #$00
EC4B   85 7F      STA $7F        drive 0
EC4D   A5 6F      LDA $6F        job for drive 0?
EC4F   F0 0B      BEQ $EC5C      no
EC51   A5 1C      LDA $1C        write protect for drive 0?
EC53   F0 03      BEQ $EC58      no
EC55   20 13 D3   JSR $D313      close all channels to drive 0
EC58   68         PLA
EC59   09 08      ORA #$08       set LED bit
EC5B   48         PHA
EC5C   E6 7F      INC $7F        increment drive number
EC5E   A5 70      LDA $70        job for drive 1?
EC60   F0 0B      BEQ $EC6D      no
EC62   A5 1D      LDA $1D        write protect for drive 1?
EC64   F0 03      BEQ $EC69      no
EC66   20 13 D3   JSR $D313      close all channels to drive 1
EC69   68         PLA
EC6A   09 00      ORA #$00
EC6C   48         PHA
EC6D   A5 86      LDA $86
EC6F   85 7F      STA $7F        get drive number back
EC71   68         PLA            bit for LED
EC72   AE 6C 02   LDX $026C      interrupt counter
EC75   F0 21      BEQ $EC98      to zero?
EC77   AD 00 1C   LDA $1C00
EC7A   E0 80      CPX #$80
```

```
EC7C    D0 03       BNE $EC81
EC7E    4C 8B EC    JMP $EC8B
EC81    AE 05 18    LDX $1805     erase timer interrupt
EC84    30 12       BMI $EC98
EC86    A2 A0       LDX #$A0
EC88    8E 05 18    STX $1805     set timer
EC8B    CE 6C 02    DEC $026C     decrement counter
EC8E    D0 08       BNE $EC98     not yet zero?
EC90    4D 6D 02    EOR $026D
EC93    A2 10       LDX #$10
EC95    8E 6C 02    STX $026C     reset counter
EC98    8D 00 1C    STA $1C00     turn LED on/off
EC9B    4C FF EB    JMP $EBFF     back to wait loop

********************************  LOAD "$"
EC9E    A9 00       LDA #$00
ECA0    85 83       STA $83       secondary address 0
ECA2    A9 01       LDA #$01
ECA4    20 E2 D1    JSR $D1E2     find channel and buffer
ECA7    A9 00       LDA #$00
ECA9    20 C8 D4    JSR $D4C8     initialize buffer pointer
ECAC    A6 82       LDX $82       channel number
ECAE    A9 00       LDA #$00
ECB0    9D 44 02    STA $0244,X   pointer to end = zero
ECB3    20 93 DF    JSR $DF93     get buffer number
ECB6    AA          TAX
ECB7    A5 7F       LDA $7F       drive number
ECB9    9D 5B 02    STA $025B,X   bring in table
ECBC    A9 01       LDA #$01      1
ECBE    20 F1 CF    JSR $CFF1     write in buffer
ECC1    A9 04       LDA #$04      4, start address $0401
ECC3    20 F1 CF    JSR $CFF1     write in buffer
ECC6    A9 01       LDA #$01      2 times 1
ECC8    20 F1 CF    JSR $CFF1
ECCB    20 F1 CF    JSR $CFF1     write in buffer as link address
ECCE    AD 72 02    LDA $0272     drive number
ECD1    20 F1 CF    JSR $CFF1     write in buffer as line number
ECD4    A9 00       LDA #$00      line number hi
ECD6    20 F1 CF    JSR $CFF1     in buffer
ECD9    20 59 ED    JSR $ED59     directory entry in buffer
ECDC    20 93 DF    JSR $DF93     get buffer number
ECDF    0A          ASL A
ECE0    AA          TAX
ECE1    D6 99       DEC $99,X     decrement buffer pointer
ECE3    D6 99       DEC $99,X
ECE5    A9 00       LDA #$00
ECE7    20 F1 CF    JSR $CFF1     0 as line end in buffer
ECEA    A9 01       LDA #$01
ECEC    20 F1 CF    JSR $CFF1     2 times 1 as link address
ECEF    20 F1 CF    JSR $CFF1
ECF2    20 CE C6    JSR $C6CE     directory entry in buffer
ECF5    90 2C       BCC $ED23     another entry?
ECF7    AD 72 02    LDA $0272     block number lo
ECFA    20 F1 CF    JSR $CFF1     in buffer
ECFD    AD 73 02    LDA $0273     block number hi
```

```
ED00    20 F1 CF   JSR $CFF1      in buffer
ED03    20 59 ED   JSR $ED59      directory entry in buffer
ED06    A9 00      LDA #$00
ED08    20 F1 CF   JSR $CFF1      zero as end marker in buffer
ED0B    D0 DD      BNE $ECEA      buffer full? no
ED0D    20 93 DF   JSR $DF93      get buffer number
ED10    0A         ASL A
ED11    AA         TAX
ED12    A9 00      LDA #$00
ED14    95 99      STA $99,X      buffer pointer to zero
ED16    A9 88      LDA #$88       set READ flag
ED18    A4 82      LDY $82        channel number
ED1A    8D 54 02   STA $0254
ED1D    99 F2 00   STA $00F2,Y    flag for channel
ED20    A5 85      LDA $85        data byte
ED22    60         RTS

********************************
ED23    AD 72 02   LDA $0272      block number lo
ED26    20 F1 CF   JSR $CFF1      write in buffer
ED29    AD 73 02   LDA $0273      block number hi
ED2C    20 F1 CF   JSR $CFF1      in buffer
ED2F    20 59 ED   JSR $ED59      'Blocks free.' in buffer
ED32    20 93 DF   JSR $DF93      get buffer number
ED35    0A         ASL A
ED36    AA         TAX
ED37    D6 99      DEC $99,X
ED39    D6 99      DEC $99,X      buffer pointer minus 2
ED3B    A9 00      LDA #$00
ED3D    20 F1 CF   JSR $CFF1
ED40    20 F1 CF   JSR $CFF1      three zeroes as program end
ED43    20 F1 CF   JSR $CFF1
ED46    20 93 DF   JSR $DF93      get buffer number
ED49    0A         ASL A          times 2
ED4A    A8         TAY
ED4B    B9 99 02   LDA $0099,Y    buffer pointer
ED4E    A6 82      LDX $82
ED50    9D 44 02   STA $0244,X    as end marker
ED53    DE 44 02   DEC $0244,X
ED56    4C 0D ED   JMP $ED0D

********************************  transmit directory line
ED59    A0 00      LDY #$00
ED5B    B9 B1 02   LDA $02B1,Y    character from buffer
ED5E    20 F1 CF   JSR $CFF1      write in output buffer
ED61    C8         INY
ED62    C0 1B      CPY #$1B       27 characters?
ED64    D0 F5      BNE $ED5B
ED66    60         RTS

********************************  get byte from buffer
ED67    20 37 D1   JSR $D137      get byte
ED6A    F0 01      BEQ $ED6D      buffer pointer zero?
ED6C    60         RTS
```

```
ED6D    85 85       STA $85        save data byte
ED6F    A4 82       LDY $82        channel number
ED71    B9 44 02    LDA $0244,Y    set end marker
ED74    F0 08       BEQ $ED7E      zero (LOAD $)?
ED76    A9 80       LDA #$80
ED78    99 F2 00    STA $00F2,Y    set READ flag
ED7B    A5 85       LDA $85        data byte
ED7D    60          RTS

ED7E    48          PHA
ED7F    20 EA EC    JSR $ECEA      create directory line in buffer
ED82    68          PLA
ED83    60          RTS

********************************   V command,  'collect'
ED84    20 D1 C1    JSR $C1D1      find drive number in input line
ED87    20 42 D0    JSR $D042      load BAM
ED8A    A9 40       LDA #$40
ED8C    8D F9 02    STA $02F9
ED8F    20 B7 EE    JSR $EEB7      create new BAM in buffer
ED92    A9 00       LDA #$00
ED94    8D 92 02    STA $0292
ED97    20 AC C5    JSR $C5AC      load directory, find 1st flag
ED9A    D0 3D       BNE $EDD9      found?
ED9C    A9 00       LDA #$00
ED9E    85 81       STA $81        sector 0
EDA0    AD 8E FE    LDA $FE85      18
EDA3    85 80       STA $80        track 18 for BAM
EDA5    20 E5 ED    JSR $EDE5      mark dir blocks as allocated
EDA8    A9 00       LDA #$00
EDAA    8D F9 02    STA $02F9
EDAD    20 FF EE    JSR $EEFF      write BAM back to disk
EDB0    4C 94 C1    JMP $C194      done, prepare disk status

********************************
EDB3    C8          INY
EDB4    B1 94       LDA ($94),Y    save track
ED86    48          PHA
EDB7    C8          INY
EDB8    B1 94       LDA ($94),Y    and sector
EDBA    48          PHA
EDBB    A0 13       LDA #$13       pointer to side-sector block
EDBD    B1 94       LDA ($94),Y
EDBF    F0 0A       BEQ $EDCB      no track following?
EDC1    85 80       STA $80        track and
EDC3    C8          INY
EDC4    B1 94       LDA ($94),Y
EDC6    85 81       STA $81        sector of 1st side-sector block
EDC8    20 E5 ED    JSR $EDE5      mark side-sector blocks as
EDCB    68          PLA            allocated
EDCC    85 81       STA $81
EDCE    68          PLA            get track and sector back
EDCF    85 80       STA $80
EDD1    20 E5 ED    JSR $EDE5      mark blocks of file as allocated
EDD4    20 04 C6    JSR $C604      read next entry in directory
```

```
EDD7    F0 C3       BEQ $ED9C       end of directory?
EDD9    A0 00       LDY #$00
EDDB    B1 94       LDA ($94),Y     file type
EDDD    30 D4       BMI $EDB3       bit 7 set, file closed?
EDDF    20 B6 C8    JSR $C8B6       file type to zero and write BAM
EDE2    4C D4 ED    JMP $EDD4

******************************      allocate file blocks in BAM
EDE5    20 5F D5    JSR $D55F       check track and sector number
EDE8    20 90 EF    JSR $EF90       allocate block in BAM
EDEB    20 75 D4    JSR $D475       read next block
EDEE    A9 00       LDA #$00
EDF0    20 C8 D4    JSR $D4C8       buffer pointer zero
EDF3    20 37 D1    JSR $D137       get byte from buffer
EDF6    85 80       STA $80         track
EDF8    20 37 D1    JSR $D137       get byte from buffer
EDFB    85 81       STA $81         sector
EDFD    A5 80       LDA $80         another block?
EDFF    D0 03       BNE $EE04       yes
EE01    4C 27 D2    JMP $D227       close channel

EE04    20 90 EF    JSR $EF90       allocate block in BAM
EE07    20 4D D4    JSR $D44D       read next block
EE0A    4C EE ED    JMP $EDEE       continue

******************************      N command, 'header'
EE0D    20 12 C3    JSR $C312       get drive number
EE10    A5 E2       LDA $E2         drive number
EE12    10 05       BPL $EE19       not clear?
EE14    A9 33       LDA #$33
EE16    4C C8 C1    JMP $C1C8       33, 'syntax error'
EE19    29 01       AND #$01
EE1B    85 7F       STA $7F         drive number
EE1D    20 00 C1    JSR $C100       turn LED on
EE20    A5 7F       LDA $7F         drive number
EE22    0A          ASL A           times 2
EE23    AA          TAX
EE24    AC 7B 02    LDY $027B       comma position
EE27    CC 74 02    CPY $0274       compare with end name
EE2A    F0 1A       BEQ $EE46       format without ID
EE2C    B9 00 02    LDA $0200,Y     first character of ID
EE2F    95 12       STA $12,X       save
EE31    B9 01 02    LDA $0201,Y     second character
EE34    95 13       STA $13,X
EE36    20 07 D3    JSR $D307       close all channels
EE39    A9 01       LDA #$01
EE3B    85 80       STA $80         track 1
EE3D    20 C6 C8    JSR $C8C6       format disk
EE40    20 05 F0    JSR $F005       erase buffer
EE43    4C 56 EE    JMP $EE56       continue as below

EE46    20 42 D0    JSR $D042       load BAM
EE49    A6 7F       LDX $7F         drive number
EE4B    BD 01 01    LDA $0101,X
EE4E    CD D5 FE    CMP $FED5       'A', marker for 1541 format
```

```
EE51    F0 03       BEQ $EE56       ok
EE53    4C 72 D5    JMP $D572       73, 'cbm dos v2.6 1541'

EE56    20 B7 EE    JSR $EEB7       create BAM
EE59    A5 F9       LDA $F9         buffer number
EE5B    A8          TAY
EE5C    0A          ASL A
EE5D    AA          TAX
EE5E    AD 88 FE    LDA $FE88       $90, start of disk name
EE61    95 99       STA $99,X       buffer pointer to name
EE63    AE 7A 02    LDX $027A
EE66    A9 1B       LDA #$1B        27
EE68    20 6E C6    JSR $C66E       write filenames in buffer
EE6B    A0 12       LDY #$12        position 18
EE6D    A6 7F       LDX $7F         drive number
EE6F    AD D5 FE    LDA $FED5       'A', 1541 format
EE72    9D 01 01    STA $0101,X
EE75    8A          TXA
EE76    0A          ASL A           times 2
EE77    AA          TAX
EE78    B5 12       LDA $12,X       ID, first character
EE7A    91 94       STA ($94),Y     in buffer
EE7C    C8          INY
EE7D    B5 13       LDA $13,X       and second character
EE7F    91 94       STA ($94),Y     in buffer
EE81    C8          INY
EE82    C8          INY
EE83    A9 32       LDA #$32        '2'
EE85    91 94       STA ($94),Y     in buffer
EE87    C8          INY
EE88    AD D5 FE    LDA $FED5       'A'  1541 format
EE8B    91 94       STA ($94),Y     in buffer
EE8D    A0 02       LDY #$02
EE8F    91 6D       STA ($6D),Y     and at position 2
EE91    AD 85 FE    LDA $FE85       18
EE94    85 80       STA $80         track number
EE96    20 93 EF    JSR $EF93       mark block as allocated
EE99    A9 01       LDA #$01        1
EE9B    85 81       STA $81         sector number
EE9D    20 93 EF    JSR $EF93       mark block as allocated
EEA0    20 FF EE    JSR $EEFF       write BAM
EEA3    20 05 F0    JSR $F005       pointer $6D/$6E to buffer, erase
                                                 buffer
EEA6    A0 01       LDY #$01
EEA8    A9 FF       LDA #$FF
EEAA    9A 6D       STA ($6D),Y     track following is zero
EEAC    20 64 D4    JSR $D464       write BAM
EEAF    C6 81       DEC $81         decrement sector number, 0
EEB1    20 60 D4    JSR $D460       read block
EEB4    4C 94 C1    JMP $C194       prepare disk status

********************************    create BAM
EEB7    20 D1 F0    JSR $F0D1
EEBA    A0 00       LDY #$00
EEBC    A9 12       LDA #$12        18
EEBE    91 6D       STA ($6D),Y     pointer to directory track
```

```
EEC0    C8          INY
EEC1    98          TYA            1
EEC2    91 6D       STA ($6D),Y    pointer to directory sector
EEC4    C8          INY
EEC5    C8          INY
EEC6    C8          INY
EEC7    A9 00       LDA #$00
EEC9    85 6F       STA $6F
EECB    85 70       STA $70        3 bytes = 24 bits for sectors
EECD    85 71       STA $71
EECF    98          TYA            byte position
EED0    4A          LSR A
EED1    4A          LSR A          divided by 4 = track number
EED2    20 4B F2    JSR $F24B      get number of sectors
EED5    91 6D       STA ($6D),Y    and in BAM
EED7    C8          INY
EED8    AA          TAX
EED9    38          SEC
EEDA    26 6F       ROL $6F
EEDC    26 70       ROL $70        create bit model
EEDE    26 71       ROL $71
EEE0    CA          DEX
EEE1    D0 F6       BNE $EED9
EEE3    B5 6F       LDA $6F,X      3 bytes
EEE5    91 6D       STA ($6D),Y    the BAM in buffer
EEE7    C8          INY
EEE8    E8          INX
EEE9    E0 03       CPX #$03
EEEB    90 F6       BCC $EEE3
EEED    C0 90       CPY #$90       position 144?
EEEF    90 D6       BCC $EEC7      no, next track
EEF1    4C 75 D0    JMP $D075      calculate number of free blocks

********************************   write BAM if needed
EEF4    20 93 DF    JSR $DF93      get buffer number
EEF7    AA          TAX
EEF8    BD 5B 02    LDA $025B,X    command for disk controller
EEFB    29 01       AND #$01
EEFD    85 7F       STA $7F        isolate drive number
EEFF    A4 7F       LDY $7F
EF01    B9 51 02    LDA $0251,Y    BAM-changed flag set?
EF04    D0 01       BNE $EF07      yes
EF06    60          RTS

EF07    A9 00       LDA #$00
EF09    99 51 02    STA $0251,Y    reset BAM-changed flag
EF0C    20 3A EF    JSR $EF3A      set buffer pointer for BAM
EF0F    A5 7F       LDA $7F        drive number
EF11    0A          ASL A          times 2
EF12    48          PHA
EF13    20 A5 F0    JSR $F0A5      verify BAM entry
EF16    68          PLA
EF17    18          CLC
EF18    69 01       ADC #$01       increment track number
EF1A    20 A5 F0    JSR $F0A5      verify BAM entry
```

```
EF1D    A5 80       LDA $80         track
EF1F    48          PHA
EF20    A9 01       LDA #$01
EF22    85 80       STA $80         track 1
EF24    0A          ASL A
EF25    0A          ASL A           times 4
EF26    85 6D       STA $6D
EF28    20 20 F2    JSR $F220       verify BAM
EF2B    E6 80       INC $80         increment track number
EF2D    A5 80       LDA $80
EF2F    CD D7 FE    CMP $FED7       and compare with max val + 1 = 36
EF32    90 F0       BCC $EF24       ok, next track
EF34    68          PLA
EF35    85 80       STA $80         get track number back
EF37    4C 8A D5    JMP $D58A       write BAM to disk

********************************    set buffer pointer for BAM
EF3A    20 0F F1    JSR $F10F       get 6 for drive 0
EF3D    AA          TAX
EF3E    20 DF F0    JSR $F0DF       allocate buffer
EF41    A6 F9       LDX $F9         buffer number
EF43    BD E0 FE    LDA $FEE0,X     buffer address, hi byte
EF46    85 6E       STA $6E
EF48    A9 00       LDA #$00        lo byte
EF4A    85 6D       STA $6D         pointer to $6D/$6E
EF4C    60          RTS

********************************    get # of free blocks for dir
EF4D    A6 7F       LDX $7F         drive number
EF4F    BD FA 02    LDA $02FA,X     number of blocks, lo
EF52    8D 72 02    STA $0272
EF55    BD FC 02    LDA $02FC,X     number of blocks, hi
EF58    8D 73 02    STA $0273       in buffer for directory
EF5B    60          RTS

********************************    mark block as free
EF5C    20 F1 EF    JSR $EFF1       set buffer pointer
EF5F    20 CF EF    JSR $EFCF       erase bit for sector in BAM
EF62    38          SEC
EF63    D0 22       BNE $EF87       block already free, then done
EF65    B1 6D       LDA ($6D),Y     bit model of BAM
EF67    1D E9 EF    ORA $EFE9       set bit X, marker for free
EF6A    91 6D       STA ($6D),Y
EF6C    20 88 EF    JSR $EF88       set flag for BAM changed
EF6F    A4 6F       LDY $6F
EF71    18          CLC
EF72    B1 6D       LDY ($6D),Y
EF74    69 01       ADC #$01        increment # of free blocks/track
EF76    91 6D       STA ($6D),Y
EF78    A5 80       LDA $80         track
EF7A    CD 85 FE    CMP $FE85       equal to 18?
EF7D    F0 3B       BEQ $EFBA       then skip
EF7F    FE FA 02    INC $02FA,X     inc # of free blocks in disk
EF82    D0 03       BNE $EF87
EF84    FE FC 02    INC $02FC,X     increment number of blocks hi
```

```
EF87    60          RTS

******************************    set flag for 'BAM changed'
EF88    A6 7F       LDX $7F       drive number
EF8A    A9 01       LDA #$01
EF8C    9D 51 02    STA $0251,X   flag = 1
EF8F    60          RTS

******************************    mark block as allocated
EF90    20 F1 EF    JSR $EFF1     set buffer pointer
EF93    20 CF EF    JSR $EFCF     erase bit for sector in BAM
EF96    F0 36       BEQ $EFCE     already allocated, then done
EF98    B1 6D       LDA ($6D),Y
EF9A    5D E9 EF    EOR $EFE9,X   erase bit for block
EF9D    91 6D       STA ($6D),Y
EF9F    20 88 EF    JSR $EF88     set flag for BAM changed
EFA2    A4 6F       LDA $6F
EFA4    B1 6D       LDA ($6D),Y
EFA6    38          SEC
EFA7    E9 01       SBC #$01      decrement # of blocks per track
EFA9    91 6D       STA ($6D),Y
EFAB    A5 80       LDA $80       track
EFAD    CD 85 FE    CMP $FE85     18?
EFB0    F0 0B       BEQ $EFBD
EFB2    BD FA 02    LDA $02FA,X   number of free blocks lo
EFB5    D0 03       BNE $EFBA
EFB7    DE FC 02    DEC $02FC,X   decrement number of free blocks
EFBA    DE FA 02    DEC $02FA,X
EFBD    BD FC 02    LDA $02FC,X   number of free blocks hi
EFC0    D0 0C       BNE $EFCE     more than 255 blocks free?
EFC2    BD FA 02    LDA $02FA,X   free blocks lo
EFC5    C9 03       CMP #$03
EFC7    B0 05       BCS $EFCE     smaller than 3?
EFC9    A9 72       LDA #$72
EFCB    20 C7 E6    JSR $E6C7     72, 'disk full'
EFCE    60          RTS

******************************    erase bit for sector in BAM entry
EFCF    20 11 F0    JSR $F011     find BAM field for this track
EFD2    98          TYA
EFD3    85 6F       STA $6F
EFD5    A5 81       LDA $81       sector
EFD7    4A          LSR A
EFD8    4A          LSR A         divide by 8
EFD9    4A          LSR A
EFDA    38          SEC
EFDB    65 6F       ADC $6F
EFDD    A8          TAY           byte number in BAM entry
EFDE    A5 81       LDA $81       sector number
EFE0    29 07       AND #$07
EFE2    AA          TAX           bit number in BAM entry
EFE3    B1 6D       LDA ($6D),Y   byte in BAM
EFE5    3D E9 EF    AND $EFE9,X   erase bit for corresponding
EFE8    60          RTS                   sector
```

```
******************************   powers of 2
EFE9    01 02 04 08 10 20 40 80

******************************   write BAM after change
EFF1    A9 FF       LDA #$FF
EFF3    2C F9 02    BIT $02F9
EFF6    F0 0C       BEQ $F004
EFF8    10 0A       BPL $F004
EFFA    70 08       BVS $F004
EFFC    A9 00       LDA #$00
EFFE    8D F9 02    STA $02F9       reset flag
F001    4C 8A D5    JMP $D58A       write block
F004    60          RTS

******************************   erase BAM buffer
F005    20 3A EF    JSR $EF3A       pointer $6D/$6E to BAM buffer
F008    A0 00       LDY #$00
F00A    98          TYA
F00B    91 6D       STA ($6D),Y     erase BAM buffer
F00D    C8          INY
F00E    D0 FB       BNE $F00B
F010    60          RTS

******************************
F011    A5 6F       LDA $6F
F013    48          PHA
F014    A5 70       LDA $70
F016    48          PHA
F017    A6 7F       LDX $7F         drive number
F019    B5 FF       LDA $FF,X
F01B    F0 05       BEQ $F022       drive zero?
F01D    A9 74       LDA #$74
F01F    20 48 E6    JSR $E648       'drive not ready'
F022    20 0F F1    JSR $F10F       get buffer number for BAM
F025    85 6F       STA $6F
F027    8A          TXA
F028    0A          ASL A
F029    85 70       STA $70
F02B    AA          TAX
F02C    A5 80       LDA $80         track
F02E    DD 9D 02    CMP $029D,X
F031    F0 0B       BEQ $F03E
F033    E8          INX
F034    86 70       STX $70
F036    DD 9D 02    CMP $029D,X
F039    F0 03       BEQ $F03E
F03B    20 5B F0    JSR $F05B
F03E    A5 70       LDA $70
F040    A6 7F       LDX $7F         drive number
F042    9D 9B 02    STA $029B,X
F045    0A          ASL A
F046    0A          ASL A           times 4
F047    18          CLC
F048    69 A1       ADC #$A1
F04A    85 6D       STA $6D
```

```
F04C    A9 02        LDA #$02
F04E    69 00        ADC #$00
F050    85 6E        STA $6E
F052    A0 00        LDY #$00
F054    58           PLA
F055    85 70        STA $70
F057    68           PLA
F058    85 6F        STA $6F
F05A    60           RTS

********************************
F05B    A6 6F        LDX $6F
F05D    20 DF F0     JSR $F0DF
F060    A5 7F        LDA $7F        drive number
F062    AA           TAX
F063    0A           ASL A
F064    1D 9B 02     ORA $029B,X
F067    49 01        EOR #$01
F069    29 03        AND #$03
F06B    85 70        STA $70
F06D    20 A5 F0     JSR $F0A5
F070    A5 F9        LDA $F9        buffer number
F072    0A           ASL A
F073    AA           TAX
F074    A5 80        LDA $80        track
F076    0A           ASL A
F077    0A           ASL A          times 4
F078    95 99        STA $99,X      equal pointer in BAM field
F07A    A5 70        LDA $70
F07C    0A           ASL A
F07D    0A           ASL A
F07E    A8           TAY
F07F    A1 99        LDA ($99,X)
F081    99 A1 02     STA $02A1,X
F084    A9 00        LDA #$00
F086    81 99        STA ($99,X)    zero in buffer
F088    F6 99        INC $99,X      increment buffer pointer
F08A    C8           INY
F08B    98           TYA
F08C    29 03        AND #$03
F08E    D0 EF        BNE $F07F
F090    A6 70        LDX $70
F092    A5 80        LDA $80        track
F094    9D 9D 02     STA $029D,X
F097    AD F9 02     LDA $02F9
F09A    D0 03        BNE $F09F
F09C    4C 8A D5     JMP $D58A      write block

F09F    09 80        ORA #$80
F0A1    8D F9 02     STA $02F9
F0A4    60           RTS

F0A5    A8           TAY
F0A6    B9 9D 02     LDA $029D,Y
F0A9    F0 25        BEQ $F0D0
```

```
F0AB    48          PHA
F0AC    A9 00       LDA #$00
F0AE    99 9D 02    STA $029D,Y
F0B1    A5 F9       LDA $F9          buffer number
F0B3    0A          ASL A            times 2
F0B4    AA          TAX
F0B5    68          PLA
F0B6    0A          ASL A
F0B7    0A          ASL A
F0B8    95 99       STA $99,X
F0BA    98          TYA
F0BB    0A          ASL A
F0BC    0A          ASL A
F0BD    A8          TAY
F0BE    B9 A1 02    LDA $02A1,Y
F0C1    81 99       STA ($99,X)      write in buffer
F0C3    A9 00       LDA #$00
F0C5    99 A1 02    STA $02A1,Y
F0C8    F6 99       INC $99,X        increment buffer pointer
F0CA    C8          INY
F0CB    98          TYA
F0CC    29 03       AND #$03
F0CE    D0 EE       BNE $F0BE
F0D0    60          RTS

F0D1    A5 7F       LDA $7F          drive number
F0D3    0A          ASL A
F0D4    AA          TAX
F0D5    A9 00       LDA #$00
F0D7    9D 9D 02    STA $029D,X
F0DA    E8          INX
F0DB    9D 9D 02    STA $029D,X
F0DE    60          RTS

F0DF    B5 A7       LDA $A7,X
F0E1    C9 FF       CMP #$FF
F0E3    D0 25       BNE $F10A
F0E5    8A          TXA
F0F6    48          PHA
F0E7    20 8E D2    JSR $D28E
F0EA    AA          TAX
F0EB    10 05       BPL $F0F2
F0ED    A9 70       LDA #$70
F0EF    20 C8 C1    JSR $C1C8        70, 'no channel'
F0F2    86 F9       STX $F9
F0F4    68          PLA
F0F5    A8          TAY
F0F6    8A          TXA
F0F7    09 80       ORA #$80
F0F9    99 A7 00    STA $00A7,Y
F0FC    0A          ASL A
F0FD    AA          TAX
F0FE    AD 85 FE    LDA $FE85        18, directory track
F101    95 06       STA $06,X        save
F103    A9 00       LDA #$00         0
```

```
F105   95 07      STA $07,X      as sector
F107   4C 86 D5   JMP $D586      write block

F10A   29 0F      AND #$0F
F10C   85 F9      STA $F9        buffer number
F10E   60         RTS

******************************** get buffer number for BAM
F10F   A9 06      LDA #$06
F111   A6 7F      LDX $7F        drive number
F113   D0 03      BNE $F118
F115   18         CLC
F116   69 07      ADC #$07       gives 13 for drive 0
F118   60         RTS

******************************** buffer number for BAM
F119   20 0F F1   JSR $F10F      get buffer number
F11C   AA         TAX
F11D   60         RTS

******************************** find and allocate free block
F11E   20 3E DE   JSR $DE3E      get track and sector number
F121   A9 03      LDA #$03
F123   85 6F      STA $6F        counter
F125   A9 01      LDA #$01
F127   0D F9 02   ORA $02F9
F12A   8D F9 02   STA $02F9
F12D   A5 6F      LDA $6F        save counter
F12F   48         PHA
F130   20 11 F0   JSR $F011      find BAM field for this track
F133   68         PLA
F134   85 6F      STA $6F        get counter back
F136   B1 6D      LDA ($6D),Y    number of free blocks in track
F138   D0 39      BNE $F173      blocks still free?
F13A   A5 80      LDA $80        track
F13C   CD 85 FE   CMP $FE85      18, directory track?
F13F   F0 19      BEQ $F15A      yes, 'disk full'
F141   90 1C      BCC $F15F      smaller, then next lower track
F143   E6 80      INC $80        increment track number
F145   A5 80      LDA $80
F147   CD D7 FE   CMP $FED7      36, highest track number plus one
F14A   D0 E1      BNE $F12D      no, continue searching this track
F14C   AE 85 FE   LDX $FE85      18, directory track
F14F   CA         DEX            decrement
F150   86 80      STX $80  .     save as track number
F152   A9 00      LDA #$00
F154   85 81      STA $81        begin with sector number zero
F156   C6 6F      DEC $6F        decrement counter
F158   D0 D3      BNE $F12D      not yet zero, then continue
F15A   A9 72      LDA #$72
F15C   20 C8 C1   JSR $C1C8      72, 'disk full'
F15F   C6 80      DEC $80        decrement track number
F161   D0 CA      BNE $F12D      not yet 0, continue in this track
F163   AE 85 FE   LDX $FE85      18, directory track
F166   E8         INX            increment
```

```
F167   86 80      STX $80       save as track number
F169   A9 00      LDA #$00
F16B   85 81      STA $81       begin with sector zero
F16D   C6 6F      DEC $6F       decrement counter
F16F   D0 BC      BNE $F12D     not yet zero, then continue
F171   F0 E7      BEQ $F15A     else 'disk full'

F173   A5 81      LDA $81       sector number
F175   18         CLC
F176   65 69      ADC $69       plus step width (10)
F178   85 81      STA $81       as new number
F17A   A5 80      LDA $80       track number
F17C   20 4B F2   JSR $F24B     get maximum sector number
F17F   8D 4E 02   STA $024E
F182   8D 4D 02   STA $024D     and save
F185   C5 81      CMP $81       greater than selected sector #?
F187   B0 0C      BCS $F195     yes
F189   38         SEC           else
F18A   A5 81      LDA $81       sector number
F18C   ED 4E 02   SBC $024E     minus maximum sector number
F18F   85 81      STA $81       save as new sector number
F191   F0 02      BEQ $F195     zero?
F193   C6 81      DEC $81       else decrement sector no. by one
F195   20 FA F1   JSR $F1FA     check BAM, find free sector
F198   F0 03      BEQ $F19D     not found?
F19A   4C 90 EF   JMP $EF90     allocate block in BAM
F19D   A9 00      LDA #$00
F19F   85 81      STA $81       sector zero
F1A1   20 FA F1   JSR $F1FA     find free sector
F1A4   D0 F4      BNE $F19A     found?
F1A6   4C F5 F1   JMP $F1F5     no, 'dir sector'

******************************* find free sector and allocate
F1A9   A9 01      LDA #$01
F1AB   0D F9 02   ORA $02F9
F1B1   A5 86      LDA $86
F1B3   48         PHA
F1B4   49 01      LDA #$01      track counter
F1B6   85 86      STA $86
F1B8   AD 85 FE   LDA $FE85     18, directory track
F1BB   38         SEC
F1BC   E5 86      SBC $86       minus counter
F1BE   85 80      STA $80       save as track number
F1C0   90 09      BCC $F1CB     result <= zero?
F1C2   F0 07      BEQ $F1CB     then try top half of dir
F1C4   20 11 F0   JSR $F011     find BAM field for this track
F1C7   B1 6D      LDA ($6D),Y   no. of free blocks in this track
F1C9   D0 1B      BNE $F1E6     free blocks exist
F1CB   AD 85 FE   LDA $FE85     18, directory track
F1CE   18         CLC
F1CF   65 86      ADC $86       plus counter
F1D1   85 80      STA $80       save as track number
F1D3   E6 86      INC $86       increment counter
F1D5   CD D7 FE   CMP $FED7     36, max track number plus one
F1D8   90 05      BCC $F1DF     smaller, then ok
```

```
F1DA    A9 67       LDA #$67
F1DC    20 45 E6    JSR $E645      67, 'illegal track or sector'
F1DF    20 11 F0    JSR $F011      find BAM field for this track
F1E2    B1 6D       LDA ($6D),Y    no. of free blocks in this track
F1E4    F0 D2       BEQ $F1BB      no more free blocks?
F1E6    68          PLA
F1E7    85 86       STA $86
F1E9    A9 00       LDA #$00
F1EB    85 81       STA $81        sector 0
F1ED    20 FA F1    JSR $F1FA      find free sector
F1F0    F0 03       BEQ $F1F5      not found?
F1F2    4C 90 EF    JMP $EF90      allocate block in BAM

F1F5    A9 71       LDA #$71
F1F7    20 45 E6    JSR $E645      71, 'dir error'

********************************   find free sectors in actual track
F1FA    20 11 F0    JSR $F011      find BAM field for this track
F1FD    98          TYA            points to # of free blocks
F1FE    48          PHA
F1FF    20 20 F2    JSR $F220      verify BAM
F202    A5 80       LDA $80        track
F204    20 4B F2    JSR $F24B      get max # of sectors of the track
F207    8D 4E 02    STA $024E      save
F20A    68          PLA
F20B    85 6F       STA $6F        save pointer
F20D    A5 81       LDA $81        compare sector
F20F    CD 4E 02    CMP $024E      with maximum number
F212    B0 09       BCS $F21D      greater than or equal to?
F214    20 D5 EF    JSR $EFD5      get bit number of sector
F217    D0 06       BNE $F21F      sector free?
F219    E6 81       INC $81        increment sector number
F21B    D0 F0       BNE $F20D      and check if free
F21D    A9 00       LDA #$00       no sectors free
F21F    60          RTS

********************************   verify no. of free blocks in BAM
F220    A5 6F       LDA $6F
F222    48          PHA
F223    A9 00       LDA #$00
F225    85 6F       STA $6F        counter to zero
F227    AC 86 FE    LDY $FE86      4, no. of bytes per track in BAM
F22A    88          DEY
F22B    A2 07       LDX #$07
F22D    B1 6D       LDA ($6D),Y
F22F    3D E9 EF    AND $EFE9,X    isolate bit
F232    F0 02       BEQ $F236
F234    E6 6F       INC $6F        increment counter of free sectors
F236    CA          DEX
F237    10 F4       BPL $F22D
F239    88          DEY
F23A    D0 EF       BNE $F22B
F23C    B1 6D       LDA ($6D),Y    comparo with number on diskette
F23E    C5 6F       CMP $6F
F240    D0 04       BNE $F246      not equal, then error
```

```
F242    68          PLA
F243    85 6F       STA $6F
F245    60          RTS
F246    A9 71       LDA #$71
F248    20 45 E6    JSR $E645      71, 'dir error'

******************************     establish # of sectors per track
F24B    AE D6 FE    LDX $FED6      4 different values
F24E    DD D6 FE    CMP $FED6,X    track number
F251    CA          DEX
F252    B0 FA       BCS $F24E      not greater?
F254    BD D1 FE    LDA $FED1,X    get number of sectors
F257    60          RTS

F258    60          RTS

******************************     initialize disk controller
F259    A9 6F       LDA #$6F       bit 4 (write prot) & 7 (SYNC)
F25B    8D 02 1C    STA $1C02      data direction register port B
F25E    29 F0       AND #$F0
F260    8D 00 1C    STA $1C00      port B, control port
F263    AD 0C 1C    LDA $1C0C      PCR, control register
F266    29 FE       AND #$FE
F268    09 0E       ORA #$0E
F26A    09 E0       ORA #$E0
F26C    8D 0C 1C    STA $1C0C
F26F    A9 41       LDA #$41
F271    8D 0B 1C    STA $1C0B      timer 1 free running, enable
F274    A9 00       LDA #$00       port A latch
F276    8D 06 1C    STA $1C06      timer 1 lo latch
F279    A9 3A       LDA #$3A
F27B    8D 07 1C    STA $1C07      timer 1 hi latch
F27E    8D 05 1C    STA $1C05      timer 1 hi
F281    A9 7F       LDA #$7F
F283    8D 0E 1C    STA $1C0E      erase IRQs
F286    A9 C0       LDA #$C0
F288    8D 0D 1C    STA $1C0D
F28B    8D 0E 1C    STA $1C0E      IER, allow interrupts
F28E    A9 FF       LDA #$FF
F290    85 3E       STA $3E
F292    85 51       STA $51        track counter for formatting
F294    A9 08       LDA #$08       8
F296    85 39       STA $39        constants for block header
F298    A9 07       LDA #$07       7
F29A    85 47       STA $47        constants for data block
F29C    A9 05       LDA #$05
F29E    85 62       STA $62
F2A0    A9 FA       LDA #$FA       pointer $62/$63 to $FA05
F2A2    85 63       STA $63
F2A4    A9 C8       LDA #$C8       200
F2A6    85 64       STA $64
F2A8    A9 04       LDA #$04
F2AA    85 5E       STA $5E
F2AC    A9 04       LDA #$04
F2AE    85 6F       STA $6F
```

```
********************************    IRQ routine for disk controller
F2B0   BA          TSX
F2B1   86 49       STX $49         save stack pointer
F2B3   AD 04 1C    LDA $1C04
F2B6   AD 0C 1C    LDA $1C0C       erase interrupt flag from timer
F2B9   09 0E       ORA #$0E
F2BB   8D 0C 1C    STA $1C0C
F2BE   A0 05       LDY #$05
F2C0   B9 00 00    LDA $0000,Y     command for buffer Y?
F2C3   10 2E       BPL $F2F3       no
F2C5   C9 D0       CMP #$D0        exec. code for program in buffer
F2C7   D0 04       BNE $F2CD       no
F2C9   98          TYA
F2CA   4C 70 F3    JMP $F370       execute program in buffer
F2CD   29 01       AND #$01        isolate drive number
F2CF   F0 07       BEQ $F2D8       drive zero?
F2D1   84 3F       STY $3F
F2D3   A9 0F       LDA #$0F        else
F2D5   4C 69 F9    JMP $F969       74, 'drive not ready'

F2D8   AA          TAX
F2D9   85 3D       STA $3D
F2DB   C5 3E       CMP $3E         motor running?
F2DD   F0 0A       BEQ $F2E9       yes
F2DF   20 7E F9    JSR $F97E       turn drive motor on
F2E2   A5 3D       LDA $3D
F2E4   85 3E       STA $3E         set flag
F2E6   4C 9C F9    JMP $F99C       to job loop

F2E9   A5 20       LDA $20
F2EB   30 03       BMI $F2F0       head transport programmed?
F2ED   0A          ASL A
F2EE   10 09       BPL $F2F9
F2F0   4C 9C F9    JMP $F99C       to job loop

F2F3   88          DEY
F2F4   10 CA       BPL $F2C0       check next buffer
F2F6   4C 9C F9    JMP $F99C       to job loop

F2F9   A9 20       LDA #$20
F2FB   85 20       STA $20         program head transport
F2FD   A0 05       LDY #$05
F2FF   84 3F       STY $3F         initialize buffer counter
F301   20 93 F3    JSR $F393       set pointer in buffer
F304   30 1A       BMI $F320       command for buffer?
F306   C6 3F       DEC $3F         decrement counter
F308   10 F7       BPL $F301       check next buffer
F30A   A4 41       LDY $41         buffer number
F30C   20 95 F3    JSR $F395       set pointer in buffer
F30F   A5 42       LDA $42         track difference for last job
F311   85 4A       STA $4A         as counter for head transport
F313   06 4A       ASL $4A
F315   A9 60       LDA #$60        set flag for head transport
F317   85 20       STA $20
```

```
F319   B1 32      LDA ($32),Y   get track number from buffer
F31B   85 22      STA $22
F31D   4C 9C F9   JMP $F99C     to job loop
F320   29 01      AND #$01      isolate drive number
F322   C5 3D      CMP $3D       equal drive number of last job?
F324   D0 E0      BNE $F306     no
F326   A5 22      LDA $22       last track number
F328   F0 12      BEQ $F33C     equal zero?
F32A   38         SEC
F32B   F1 32      SBC ($32),Y   equal track number of this job?
F32D   F0 0D      BEQ $F33C     yes
F32F   49 FF      EOR #$FF
F331   85 42      STA $42
F333   E6 42      INC $42
F335   A5 3F      LDA $3F       drive number
F337   85 41      STA $41
F339   4C 06 F3   JMP $F306

E33C   A2 04      LDX #$04
F33E   B1 32      LDA ($32),Y   track number of the job
F340   85 40      STA $40       save
F342   DD D6 FE   CMP $FED6,X   compare with max track number
F345   CA         DEX
F346   B0 FA      BCS $F342     greater?
F348   8D D1 FE   LDA $FED1,X   get # of sectors per track
F34B   85 43      STA $43       and save
F34D   8A         TXA
F34E   0A         ASL A
F34F   0A         ASL A
F350   0A         ASL A
F351   0A         ASL A
F352   0A         ASL A
F353   85 44      STA $44       gives 0, 32, 64, 96
F355   AD 00 1C   LDA $1C00
F358   29 9F      AND #$9F
F35A   05 44      ORA $44       generate control byte for notor
F35C   8D 00 1C   STA $1C00
F35F   A6 3D      LDX $3D
F361   A5 45      LDA $45       command code
F363   C9 40      CMP #$40      position head?
F365   F0 15      BEQ $F37C     yes
F367   C9 60      CMP #$60      command code for prg execution?
F369   F0 03      BEQ $F36E     yes
F36B   4C B1 F3   JMP $F3B1     read block header

********************************   execute program in buffer
F36E   A5 3F      LDA $3F       buffer number
F370   18         CLC
F371   69 03      ADC #$03      plus 3
F373   85 31      STA $31
F375   A9 00      LDA #$00      equals address of buffer
F377   85 30      STA $30
F379   6C 30 00   JMP ($0030)   execute program in buffer

********************************   position head
```

```
F37C    A9 60       LDA #$60
F37E    85 20       STA $20        set flag for head transport
F380    AD 00 1C    LDA $1C00
F383    29 FC       AND #$FC       turn stepper motors on
F385    8D 00 1C    STA $1C00
F388    A9 A4       LDA #$A4       164
F38A    85 4A       STA $4A        step counter for head transport
F38C    A9 01       LDA #$01
F38E    85 22       STA $22        track number
F390    4C 69 F9    JMP $F969      ok

******************************     initialize pointer in buffer
F393    A4 3F       LDY $3F        buffer number
F395    B9 00 00    LDA $0000,Y    command code
F398    48          PHA            save
F399    10 10       BPL $F3AB
F39B    29 78       AND #$78       erase bits 0,1,2, and 7
F39D    85 45       STA $45
F39F    98          TYA            buffer number
F3A0    0A          ASL A          times two
F3A1    69 06       ADC #$06       plus 6
F3A3    85 32       STA $32        equals pointer to actual buffer
F3A5    98          TYA            buffer number
F3A6    18          CLC
F3A7    69 03       ADC #$03       plus 3
F3A9    85 31       STA $31        equals buffer address hi
F3AB    A0 00       LDY #$00
F3AD    84 30       STY $30        buffer address lo
F3AF    68          PLA            get command code back
F3B0    60          RTS

******************************     read block header, verify ID
F3B1    A2 5A       LDX #$5A       90
F3B3    86 4B       STX $4B        counter
F3B5    A2 00       LDX #$00
F3B7    A9 52       LDA #$52       82
F3B9    85 24       STA $24s
F3BB    20 56 F5    JSR $F556      wait for SYNC
F3BE    50 FE       BVC $F3BE      byte ready?
F3C0    B8          CLV
F3C1    AD 01 1C    LDA $1C01      data from read head
F3C4    C5 24       CMP $24
F3C6    D0 3F       BNE $F407      20, 'read error'
F3C8    50 FE       BVC $F3C8      byte ready?
F3CA    B8          CLV
F3CB    AD 01 1C    LDA $1C01      data byte from disk(block header)
F3CE    95 25       STA $25,X      save 7 bytes
F3D0    E8          INX
F3D1    E0 07       CPX #$07
F3D3    D0 F3       BNE $F3C8      continue reading
F3D5    20 97 F4    JSR $F497
F3D8    A0 04       LDY #$04       4 bytes plus parity
F3DA    A9 00       LDA #$00
F3DC    59 16 00    EOR $0016,Y    form checksum for header
F3DF    88          DEY
```

```
F3E0    10 FA       BPL $F3DC
F3E2    C9 00       CMP #$00       parity ok?
F3E4    D0 38       BNE $F41E      27, 'read error'
F3E6    A6 3E       LDX $3E        drive number
F3E8    A4 18       LDA $18        track number of header
F3EA    95 22       STA $22,X      use as actual track number
F3EC    A5 45       LDA $45
F3EE    C9 30       CMP #$30       code for 'preserve header'
F3F0    F0 1E       BEQ $F410      preserve header
F3F2    A5 3E       LDA $3E
F3F4    0A          ASL A
F3F5    A8          TAY
F3F6    B9 12 00    LDA $0012,Y
F3F9    C5 16       CMP $16        compare with ID1
F3FB    D0 1E       BNE $F41B
F3FD    B9 13 00    LDA $0013,Y
F400    C5 17       CMP $17        compare with ID2
F402    D0 17       BNE $F41B      <>, then 29, 'disk id mismatch'
F404    4C 23 F4    JMP $F423

F407    C6 4B       DEC $4B        decrement counter for attempts
F409    D0 B0       BNE $F3BB      and try again
F40B    A9 02       LDA #$02       else
F40D    20 69 F9    JSR $F969      20, 'read error'

********************************   preserve block header
F410    A5 16       LDA $16        ID1
F412    85 12       STA $12
F414    A5 17       LDA $17        and ID2
F416    85 13       STA $13        preserve
F418    A9 01       LDA #$01       ok
F41A    2C          .BYTE $2C
F41B    A9 0B       LDA #$0B       29, 'disk id mismatch'
F41D    2C          .BYTE $2C
F41E    A9 09       LDA #$09       27, 'write error'
F420    4C 69 F9    JMP $F969      done

********************************
F423    A9 7F       LDA #$7F
F425    85 4C       STA $4C
F427    A5 19       LDA $19
F429    18          CLC
F42A    69 02       ADC #$02
F42C    C5 43       CMP $43
F42E    90 02       BCC $F432
F430    E5 43       SBC $43
F432    85 4D       STA $4D
F434    A2 05       LDX #$05
F436    86 3F       STX $3F
F438    A2 FF       LDX #$FF
F43A    20 93 F3    JSR $F393      set buffer ptr for disk control
F43D    10 44       BPL $F483
F43F    85 44       STA $44
F441    29 01       AND #$01
F443    C5 3E       CMP $3E
```

```
F445    D0 3C       BNE $F483
F447    A0 00       LDY #$00
F449    B1 32       LDA ($32),Y
F44B    C5 40       CMP $40
F44D    D0 34       BNE $F483
F44F    A5 45       LDA $45        command code
F451    C9 60       CMP #$60
F453    F0 0C       BEQ $F461
F455    A0 01       LDY #$01
F457    38          SEC
F458    B1 32       LDA ($32),Y
F45A    E5 4D       SBC $4D
F45C    10 03       BPL $F461
F45E    18          CLC
F45F    65 43       ADC $43
F461    C4 4C       CMP $4C
F463    B0 1E       BCS $F483
F465    48          PHA
F466    A5 45       LDA $45
F468    F0 14       BEQ $F47E
F46A    68          PLA
F46B    C9 09       CMP #$09
F46D    90 14       BCC $F483
F46F    C9 0C       CMP #$0C
F471    B0 10       BCS $F483
F473    85 4C       STA $4C
F475    A5 3F       LDA $3F
F477    AA          TAX
F478    69 03       ADC #$03
F47A    85 31       STA $31
F47C    D0 05       BNE $F483
F47E    68          PLA
F47F    C9 06       CMP #$06
F481    90 F0       BCC $F473
F483    C6 3F       DEC $3F
F485    10 B3       BPL $F43A
F487    8A          TXA
F488    10 03       BPL $F48D
F48A    4C 9C F9    JMP $F99C      to job loop

F48D    86 3F       STX $3F
F48F    20 93 F3    JSR $F393      get buffer number
F492    A5 45       LDA $45        command code
F494    4C CA F4    JMP $F4CA      continue checking

F497    A5 30       LDA $30
F499    48          PLA            save pointer $30/$31
F49A    A5 31       LDA $31
F49C    48          PHA
F49D    A9 24       LDA #$24
F49F    85 30       STA $30
F4A1    A9 00       LDA #$00       pointer $30/$31 to $24
F4A3    85 31       STA $31
F4A5    A9 00       LDA #$00
F4A7    85 34       STA $34
```

```
F4A9   20 E6 F7   JSR $F7E6
F4AC   A5 55      LDA $55
F4AE   85 18      STA $18
F4B0   A5 54      LDA $54
F4B2   85 19      STA $19
F4B4   A5 53      LDA $53
F4B6   85 1A      STA $1A
F4B8   20 E6 F7   JSR $F7E6
F4BB   A5 52      LDA $52
F4BD   85 17      STA $17
F4BF   A5 53      LDA $53
F4C1   85 16      STA $16
F4C3   68         PLA
F4C4   85 31      STA $31
F4C6   68         PLA          get pointer $30/$31 back
F4C7   85 30      STA $30
F4C9   60         RTS

*******************************
F4CA   C9 00      CMP #$00     command code for 'read'?
F4CC   F0 03      BEQ $F4D1    yes
F4CE   4C 6E F5   JMP $F56E    continue checking command code

F4D1   20 0A F5   JSR $F50A    find beginning of data block
F4D4   50 FE      BVC $F4D4    byte ready?
F4D6   B8         CLV
F4D7   AD 01 1C   LDA $1C01    get data byte
F4DA   91 30      STA ($30),Y  and write in buffer
F4DC   C8         INY          256 times
F4DD   D0 F5      BNE $F4D4
F4DF   A0 BA      LDY #$BA
F4E1   50 FE      BVC $F4E1    byte ready?
F4E3   B8         CLV
F4E4   AD 01 1C   LDA $1C01    read bytes
F4E7   99 00 01   STA $0100,Y  from $1BA to $1FF
F4EA   C8         INY
F4EB   D0 F4      BNE $F4E1
F4ED   20 E0 F8   JSR $F8E0
F4F0   A5 38      LDA $38
F4F2   C5 47      CMP $47      equal 7, beginning of data block?
F4F4   F0 05      BEQ $F4FB    yes
F4F6   A9 04      LDA #$04     22, 'read error'
F4F8   4C 69 F9   JMP $F969    error termination

F4FB   20 E9 F5   JSR $F5E9    calculate parity of data block
F4FE   C5 3A      CMP $3A      agreement?
F500   F0 03      BEQ $F505    yes
F502   A9 05      LDA #$05     23, 'read error'
F504   2C         .BYTE $2C
F505   A9 01      LDA #$01     ok
F507   4C 69 F9   JMP $F969    prepare error message

*******************************  find start of data block
F50A   20 10 F5   JSR $F510    read block header
F50D   4C 56 F5   JMP $F556    wait for SYNC
```

```
********************************    read block header
F510   A5 3D      LDA $3D          drive number
F512   0A         ASL A
F513   AA         TAX
F514   B5 12      LDA $12,X        ID1
F516   85 16      STA $16          save
F518   B5 13      LDA $13,X        ID2
F51A   85 17      STA $17          save
F51C   A0 00      LDY #$00
F51E   B1 32      LDA ($32),Y      get track and
F520   85 18      STA $18
F522   C8         INY
F523   B1 32      LDA ($32),Y      sector number from buffer
F525   85 19      STA $19
F527   A9 00      LDA #$00
F529   45 16      EOR $16
F52B   45 17      EOR $17          calculate parity for block header
F52D   45 18      EOR $18
F52F   45 19      EOR $19
F531   85 1A      STA $1A          and save
F533   20 34 F9   JSR $F934
F536   A2 5A      LDX #$5A         90 attempts
F538   20 56 F5   JSR $F556        wait for SYNC
F53B   A0 00      LDY #$00
F53D   50 FE      BVC $F35D        byte ready?
F53F   B8         CLV
F540   AD 01 1C   LDA $1C01        read data from block header
F543   D9 24 00   CMP $0024,Y      compare with saved data
F546   D0 06      BNE $F54E        not the same, try again
F548   C8         INY
F549   C0 08      CPY #$08         8 bytes read?
F54B   D0 F0      BNE $F53D        no
F54D   60         RTS

F54E   CA         DEX              decrement counter
F54F   D0 E7      BNE $F538        not yet zero?
F551   A9 02      LDA #$02
F553   4C 69 F9   JMP $F969        20, 'read error'

********************************    wait for SYNC
F556   A9 D0      LDA #$D0         208
F558   8D 05 18   STA $1805        start timer
F55B   A9 03      LDA #$03         error code
F55D   2C 05 18   BIT $1805
F560   10 F1      BPL $F553        timer run down, then 'read error'
F562   2C 00 1C   BIT $1C00        SYNC signal
F565   30 F6      BMI $F55D        not yet found?
F567   AD 10 1C   LDA $1C01        read byte
F56A   B8         CLV
F56B   A0 00 •    LDY #$00
F56D   60         RTS

********************************
F56E   C9 10      CMP #$10         command code for 'write'
```

```
F570    F0 03       BEQ $F575      yes
F572    4C 91 F6    JMP $F691      continue checking command code

******************************     write data block to disk
F575    20 E9 F5    JSR $F5E9      calculate parity for buffer
F57B    85 3A       STA $3A        and save
F57A    AD 00 1C    LDA $1C00      read port B
F57D    29 10       AND #$10       isolate bit for 'write protect'
F57F    D0 05       BNE $F586      not set, ok
F581    A9 08       LDA #$08
F583    4C 69 F9    JMP $F969      26, 'write protect'

F586    20 8F F7    JSR $F78F
F589    20 10 F5    JSR $F510      find block header
F58C    A2 09       LDX #$09
F58E    50 FE       BVC $F58E      byte ready?
F590    B8          CLV
F591    CA          DEX
F592    D0 FA       BNE $F58E
F594    A9 FF       LDA #$FF
F596    8D 03 1C    STA $1C03      port A (write/read head) to
F599    AD 0C 1C    LDA $1C0C      to output
F59C    29 1F       AND #$1F
F59E    09 C0       ORA #$C0       change PCR to output
F5A0    8D 0C 1C    STA $1C0C
F5A3    A9 FF       LDA #$FF
F5A5    A2 05       LDX #$05
F5A7    8D 01 1C    STA $1C01      write $FF to disk 5 times
F5AA    B8          CLV
F5AB    50 FE       BVC $F5AB      as SYNC characters
F5AD    B8          CLV
F5AE    CA          DEX
F5AF    D0 FA       BNE $F5AB
F5B1    A0 BB       LDY #$BB
F5B3    B9 00 01    LDA $0100,Y    bytes $1BB to $1FF to disk
F5B6    50 FE       BVC $F5B6
F5B8    B8          CLV
F5B9    8D 01 1C    STA $1C01
F5BC    C8          INY
F5BD    D0 F4       BNE $F5B3
F5BF    B1 30       LDA ($30),Y    write data buffer (256 bytes)
F5C1    50 FE       BVC $F5C1
F5C3    B8          CLV
F5C4    8D 01 1C    STA $1C01
F5C7    C8          INY
F5C8    D0 F5       BNE $F5BF
F5CA    50 FE       BVC $F5CA      byte ready?
F5CC    AD 0C 1C    LDA $1C0C
F5CF    09 E0       ORA #$E0       PCR to input again
F5D1    8D 0C 1C    STA $1C0C
F5D4    A9 00       LDA #$00
F5D6    8D 03 1C    LDA $1C03      port A (read/write head) to input
F5D9    20 F2 F5    JSR $F5F2
F5DC    A4 3F       LDY $3F
F5DE    B9 00 00    LDA $0000,Y
```

```
F5E1    49 30       EOR #$30       convert command code 'write'
F5E3    99 00 00    STA $0000,Y          to 'verify'
F5E6    4C B1 F3    JMP $F3B1

******************************     calculate parity for data buffer
F5E9    A9 00       LDA #$00
F5EB    A8          TAY
F5EC    51 30       EOR ($30),Y
F5EE    C8          INY
F5EF    D0 FB       BNE $F5EC
F5F1    60          RTS

F5F2    A9 00       LDA #$00
F5F4    85 2E       STA $2E
F5F6    85 30       STA $30
F5F8    85 4F       STA $4F
F5FA    A5 31       LDA $31
F5FC    85 4E       STA $4E
F5FE    A9 01       LDA #$01
F600    85 31       STA $31
F602    85 2F       STA $2F
F604    A9 BB       LDA #$BB
F606    85 34       STA $34
F608    85 36       STA $36
F60A    20 E6 F7    JSR $F7E6
F60D    A5 52       LDA $52
F60F    85 38       STA $38
F611    A4 36       LDY $36
F613    A5 53       LDA $53
F615    91 2E       STA ($2E),Y
F617    C8          INY
F618    A5 54       LDA $54
F61A    91 2E       STA ($2E),Y
F61C    C8          INY
F61D    A5 55       LDA $55
F61F    91 2E       STA ($2E),Y
F621    C8          INY
F622    84 36       STY $36
F624    20 E6 F7    JSR $F7E6
F627    A4 36       LDY $36
F629    A5 52       LDA $52
F62B    91 2E       STA ($2E),Y
F62D    C8          INY
F62E    A5 53       LDA $53
F630    91 2E       STA ($2E),Y
F632    C8          INY
F633    F0 0E       BEQ $F643
F635    A5 54       LDA $54
F637    91 2E       STA ($2E),Y
F639    C8          INY
F63A    A5 55       LDA $55
F63C    91 2E       STA ($2E),Y
F63E    C8          INY
F63F    84 36       STY $36
F641    D0 E1       BNE $F624
```

```
F643    A5 54       LDA $54
F645    91 30       STA ($30),Y
F647    C8          INY
F648    A5 55       LDA $55
F64A    91 30       STA ($30),Y
F64C    C8          INY
F64D    84 36       STY $36
F64F    20 E6 F7    JSR $F7E6
F652    A4 36       LDY $36
F654    A5 52       LDA $52
F656    91 30       STA ($30),Y
F658    C8          INY
F659    A5 53       LDA $53
F65B    91 30       STA ($30),Y
F65D    C8          INY
F65E    A5 54       LDA $54
F660    91 30       STA ($30),Y
F662    C8          INY
F663    A5 55       LDA $55
F665    91 30       STA ($30),Y
F667    C8          INY
F668    84 36       STY $36
F66A    C0 BB       CPY #$BB
F66C    90 E1       BCC $F64F
F66E    A9 45       LDA #$45
F670    85 2E       STA $2E
F672    A5 31       LDA $31
F674    85 2F       STA $2F
F676    A0 BA       LDY #$BA
F678    B1 30       LDA ($30),Y
F67A    91 2E       STA ($2E),Y
F67C    88          DEY
F67D    D0 F9       BNE $F678
F67F    B1 30       LDA ($30),Y
F681    91 2E       STA ($2E),Y
F683    A2 BB       LDX #$BB
F685    BD 00 01    LDA $0100,X
F688    91 30       STA ($30),Y
F68A    C8          INY
F68B    E8          INX
F68C    D0 F7       BNE $F685
F68E    86 50       STX $50
F690    60          RTS

********************************
F691    C9 20       CMP #$20        command code for 'verify'?
F693    F0 03       BEQ $F698       yes
F695    4C CA F6    JMP $F6CA       continue checking command code

F698    20 E9 F5    JSR $F5E9       calculate parity for data buffer
F69B    85 3A       STA $3A         and save
F69D    20 8F F7    JSR $F78F
F6A0    20 0A F5    JSR $F50A       find start of data block
F6A3    A0 BB       LDY #$BB
F6A5    B9 00 01    LDA $0100,Y     data from buffer
```

```
F6A8    50 FE       BVC $F6A8      byte ready?
F6AA    B8          CLV
F6AB    4D 01 1C    EOR $1C01      compare with data from disk
F6AE    D0 15       BNE $F6C5      not equal, then error
F6B0    C8          INY
F6B1    D0 F2       BNE $F6A5
F6B3    B1 30       LDA ($30),Y    data from buffer
F6B5    50 FE       BVC $F6B5
F6B7    B8          CLV
F6B8    4D 01 1C    EOR $1C01      compare with data from disk
F6BB    D0 08       BNE $F6C5      not equal, then error
F6BD    C8          INY
F6BE    C0 FD       CPY #$FD
F6C0    D0 F1       BNE $F6B3
F6C2    4C 18 F4    JMP $F418      error free termination

F6C5    A9 07       LDA #$07
F6C7    4C 69 F9    JMP $F969      25, 'write error'

********************************
F6CA    20 10 F5    JSR $F510      read block header
F6CD    4C 18 F4    JMP $F418      done

********************************
F6D0    A9 00       LDA #$00
F6D2    85 57       STA $57
F6D4    85 5A       STA $5A
F6D6    A4 34       LDY $34
F6D8    A5 52       LDA $52
F6DA    29 F0       AND #$F0       isolate hi-nibble
F6DC    4A          LSR A
F6DD    4A          LSR A          and rotate to lower nibble
F6DE    4A          LSR A
F6DF    4A          LSR A
F6E0    AA          TAX            as index in table
F6E1    BD 7F F7    LDA $F77F,X
F6E4    0A          ASL A
F6E5    0A          ASL A          times 8
F6E6    0A          ASL A
F6E7    85 56       STA $56
F6E9    A5 52       LDA $52
F6EB    29 0F       AND #$0F       isolate lower nibble
F6ED    AA          TAX            as index in table
F6EE    BD 7F F7    LDA $F77F,X
F6F1    6A          ROR A
F6F2    66 57       ROR $57
F6F4    6A          ROR A
F6F5    66 57       ROR $57
F6F7    29 07       AND #$07
F6F9    05 56       ORA $56
F6FB    91 30       STA ($30),Y    in buffer
F6FD    C8          INY            increment buffer
F6FE    A5 53       LDA $53
F700    29 F0       AND #$F0       isolate upper nibble
F702    4A          LSR A
```

```
F703    4A          LSR A
F704    4A          LSR A          shift to upper nibble
F705    4A          LSR A
F706    AA          TAX            as index in table
F707    BD 7F F7    LDA $F77F,X
F70A    0A          ASL A
F70B    05 57       ORA $57
F70D    85 57       STA $57
F70F    A5 53       LDA $53
F711    29 0F       AND #$0F       lower nibble
F713    AA          TAX            as index
F714    BD 7F F7    LDA $F77F,X
F717    2A          ROL A
F718    2A          ROL A
F719    2A          ROL A
F71A    2A          ROL A
F71B    85 58       STA $58
F71D    2A          ROL A
F71E    29 01       AND #$01
F720    05 57       ORA $57
F722    91 30       STA ($30),Y    in buffer
F724    C8          INY            increment buffer
F725    A5 54       LDA $54
F727    29 F0       AND #$F0       isolate hi-nibble
F729    4A          LSR A
F72A    4A          LSR A
F72B    4A          LSR A
F72C    4A          LSR A
F72D    AA          TAX
F72E    BD 7F F7    LDA $F77F,X
F731    18          CLC
F732    6A          ROR A
F733    05 58       ORA $58
F735    91 30       STA ($30),Y    in buffer
F737    C8          INY            increment buffer pointer
F738    6A          ROR A
F739    29 80       AND #$80
F73B    85 59       STA $59
F73D    A5 54       LDA $54
F73F    29 0F       AND #$0F       lower nibble
F741    AA          TAX            as index.
F742    BD 7F F7    LDA $F77F,X
F745    0A          ASL A
F746    0A          ASL A
F747    29 7C       AND #$7C
F749    05 59       ORA $59
F74B    85 59       STA $59
F74D    A5 55       LDA $55
F74F    29 F0       AND #$F0       isolate hi-nibble
F751    4A          LSR A
F752    4A          LSR A          shift to lower nibble
F753    4A          LSR A
F754    4A          LSR A
F755    AA          TAX            as index in table
F756    BD 7F F7    LDA $F77F,X
```

```
F759    6A          ROR A
F75A    66 5A       ROR $5A
F75C    6A          ROR A
F75D    66 5A       ROR $5A
F75F    6A          ROR A
F760    66 5A       ROR $5A
F762    29 03       AND #$03
F764    05 59       ORA $59
F766    91 30       STA ($30),Y     in buffer
F768    C8          INY             increment buffer pointer
F769    D0 04       BNE $F76F
F76B    A5 2F       LDA $2F
F76D    85 31       STA $31
F76F    A5 55       LDA $55
F771    29 0F       AND #$0F        lower nibble
F773    AA          TAX             as index
F774    BD 7F F7    LDA $F77F,X
F777    05 5A       ORA $5A
F779    91 30       STA ($30),Y     in buffer
F77B    C8          INY             increment buffer pointer
F77C    84 34       STY $34         and save
F77E    60          RTS

********************************
F77F 0A 0B 12 13 0E 0F 16 17
F787 09 19 1A 1B 0D 1D 1E 15

********************************
F78F    A9 00       LDA #$00
F791    85 30       STA $30
F793    85 2E       STA $2E
F795    85 36       STA $36
F797    A9 BB       LDA #$BB
F799    85 34       STA $34
F79B    85 50       STA $50
F79D    A5 31       LDA $31
F79F    85 2F       STA $2F
F7A1    A9 01       LDA #$01
F7A3    85 31       STA $31
F7A5    A5 47       LDA $47
F7A7    85 52       STA $52
F7A9    A4 36       LDY $36
F7AB    B1 2E       LDA ($2E),Y
F7AD    85 53       STA $53
F7AF    C8          INY
F7B0    B1 2E       LDA ($2E),Y
F7B2    85 54       STA $54
F7B4    C8          INY
F7B5    B1 2E       LDA ($2E),Y
F7B7    85 55       STA $55
F7B9    C8          INY
F7BA    84 36       STY $36
F7BC    20 D0 F6    JSR $F6D0
F7BF    A4 36       LDY $36
F7C1    B1 2E       LDA ($2E),Y
```

```
F7C3    85 52       STA $52
F7C5    C8          INY
F7C6    F0 11       BEQ $F7D9
F7C8    B1 2E       LDA ($2E),Y
F7CA    85 53       STA $53
F7CC    C8          INY
F7CD    B1 2E       LDA ($2E),Y
F7CF    85 54       STA $54
F7D1    C8          INY
F7D2    B1 2E       LDA ($2E),Y
F7D4    85 55       STA $55
F7D6    C8          INY
F7D7    D0 E1       BNE $F7BA
F7D9    A5 3A       LDA $3A
F7DB    85 53       STA $53
F7DD    A9 00       LDA #$00
F7DF    85 54       STA $54
F7E1    85 55       STA $55
F7E3    4C D0 F6    JMP $F6D0

F7E6    A4 34       LDY $34
F7E8    B1 30       LDA ($30),Y
F7EA    29 F8       AND #$F8
F7EC    4A          LSR A
F7ED    4A          LSR A
F7EE    4A          LSR A
F7EF    85 56       STA $56
F7F1    B1 30       LDA ($30),Y
F7F3    29 07       AND #$07
F7F5    0A          ASL A
F7F6    0A          ASL A
F7F7    85 57       STA $57
F7F9    C8          INY
F7FA    D0 06       BNE $F802
F7FC    A5 4E       LDA $4E
F7FE    85 31       STA $31
F800    A4 4F       LDY $4F
F802    B1 30       LDA ($30),Y
F804    29 C0       AND #$C0
F806    2A          ROL A
F807    2A          ROL A
F808    2A          ROL A
F809    05 57       ORA $57
F80B    85 57       STA $57
F80D    B1 30       LDA ($30),Y
F80F    29 3E       AND #$3E
F811    4A          LSR A
F812    85 58       STA $58
F814    B1 30       LDA ($30),Y
F816    29 01       AND #$01
F818    0A          ASL A
F819    0A          ASL A
F81A    0A          ASL A
F81B    0A          ASL A
F81C    85 59       STA $59
```

```
F81E    C8          INY
F81F    B1 30       LDA ($30),Y
F821    29 F0       AND #$F0
F823    4A          LSR A
F824    4A          LSR A
F825    4A          LSR A
F826    4A          LSR A
F827    05 59       ORA $59
F829    85 59       STA $59
F82B    B1 30       LDA ($30),Y
F82D    29 0F       AND #$0F
F82F    0A          ASL A
F830    85 5A       STA $5A
F832    C8          INY
F833    B1 30       LDA ($30),Y
F835    29 80       AND #$80
F837    18          CLC
F838    2A          ROL A
F839    2A          ROL A
F83A    29 01       AND #$01
F83C    05 5A       ORA $5A
F83E    85 5A       STA $5A
F840    B1 30       LDA ($30),Y
F842    29 7C       AND #$7C
F844    4A          LSR A
F845    4A          LSR A
F846    85 5B       STA $5B
F848    B1 30       LDA ($30),Y
F84A    29 03       AND #$03
F84C    0A          ASL A
F84D    0A          ASL A
F84E    0A          ASL A
F84F    85 5C       STA $5C
F851    C8          INY
F852    D0 06       BNE $F85A
F854    A5 4E       LDA $4E
F856    85 31       STA $31
F858    A4 4F       LDY $4F
F85A    B1 30       LDA ($30),Y
F85C    29 E0       AND #$E0
F85E    2A          ROL A
F85F    2A          ROL A
F860    2A          ROL A
F861    2A          ROL A
F862    05 5C       ORA $5C
F864    85 5C       STA $5C
F866    B1 30       LDA ($30),Y
F868    29 1F       AND #$1F
F86A    85 5D       STA $5D
F86C    C8          INY
F86D    84 34       STY $34
F86F    A6 56       LDX $56
F871    BD A0 F8    LDA $F8A0,X
F874    A6 57       LDX $57
F876    1D C0 F8    ORA $F8C0,X
```

```
F879    85 52       STA $52
F87B    A6 58       LDX $58
F87D    BD A0 F8    LDA $F8A0,X
F880    A6 59       LDX $59
F882    1D C0 F8    ORA $F8C0,X
F885    85 53       STA $53
F887    A6 5A       LDX $5A
F889    BD A0 F8    LDA $F8A0,X
F88C    A6 5B       LDX $5B
F88E    1D C0 F8    ORA $F8C0,X
F891    85 54       STA $54
F893    A6 5C       LDX $5C
F895    BD A0 F8    LDA $F8A0,X
F898    A6 5D       LDX $5D
F89A    1D C0 F8    ORA $F8C0,X
F89D    85 55       STA $55
F89F    60          RTS

********************************

F8A0 FF FF FF FF FF FF FF FF
F8A8 FF 80 00 10 FF C0 40 50
F8B0 FF FF 20 30 FF F0 60 70
F8B8 FF 90 A0 B0 FF D0 E0 FF

F8C0 FF FF FF FF FF FF FF FF
F8C8 FF 08 00 01 FF 0C 04 05
F8D0 FF FF 02 03 FF 0F 06 07
F8D8 FF 09 0A 0B FF 0D 0E FF

********************************

F8E0    A9 00       LDA #$00
F8E2    85 34       STA $34
F8E4    85 2E       STA $2E
F8E6    85 36       STA $36
F8E8    A9 01       LDA #$01
F8EA    85 4E       STA $4E
F8EC    A9 BA       LDA #$BA
F8EE    85 4F       STA $4F
F8F0    A5 31       LDA $31
F8F2    85 2F       STA $2F
F8F4    20 E6 F7    JSR $F7E6
F8F7    A5 52       LDA $52
F8F9    85 38       STA $38
F8FB    A4 36       LDY $36
F8FD    A5 53       LDA $53
F8FF    91 2E       STA ($2E),Y
F901    C8          INY
F902    A5 54       LDA $54
F904    91 2E       STA ($2E),Y
F906    C8          INY
F907    A5 55       LDA $55
F909    91 2E       STA ($2E),Y
F90B    C8          INY
F90C    84 36       STY $36
F90E    20 E6 F7    JSR $F7E6
```

```
F911    A4 36       LDY $36
F913    A5 52       LDA $52
F915    91 2E       STA ($2E),Y
F917    C8          INY
F918    F0 11       BEQ $F92B
F91A    A5 53       LDA $53
F91C    91 2E       STA ($2E),Y
F91E    C8          INY
F91F    A5 54       LDA $54
F921    91 2E       STA ($2E),Y
F923    C8          INY
F924    A5 55       LDA $55
F926    91 2E       STA ($2E),Y
F928    C8          INY
F929    D0 E1       BNE $F90C
F92B    A5 53       LDA $53
F92D    85 3A       STA $3A
F92F    A5 2F       LDA $2F
F931    85 31       STA $31
F933    60          RTS

F934    A5 31       LDA $31
F936    85 2F       STA $2F
F938    A9 00       LDA #$00
F93A    85 31       STA $31
F93C    A9 24       LDA #$24
F93E    85 34       STA $34
F940    A5 39       LDA $39
F942    85 52       STA $52
F944    A5 1A       LDA $1A
F946    85 53       STA $53
F948    A5 19       LDA $19
F94A    85 54       STA $54
F94C    A5 18       LDA $18
F94E    85 55       STA $55
F950    20 D0 F6    JSR $F6D0
F953    A5 17       LDA $17
F955    85 52       STA $52
F957    A5 16       LDA $16
F959    85 53       STA $53
F95B    A9 00       LDA #$00
F95D    85 54       STA $54
F95F    85 55       STA $55
F961    20 D0 F6    JSR $F6D0
F964    A5 2F       LDA $2F
F966    85 31       STA $31
F968    60          RTS

F969    A4 3F       LDY $3F
F96B    99 00 00    STA $0000,Y
F96E    A5 50       LDA $50
F970    F0 03       BEQ $F975
F972    20 F2 F5    JSR $F5F2
F975    20 8F F9    JSR $F98F
F978    A6 49       LDX $49        get stack pointer back
```

```
F97A    9A          TXS
F97B    4C BE F2    JMP $F2BE

F97E    A9 A0       LDA #$A0
F980    85 20       STA $20
F982    AD 00 1C    LDA $1C00
F985    09 04       ORA #$04      turn drive motor off
F987    8D 00 1C    STA $1C00
F98A    A9 3C       LDA $3C
F98C    85 48       STA $48
F98E    60          RTS

F98F    A6 3E       LDX $3E
F991    A5 20       LDA $20
F993    09 10       ORA #$10
F995    85 20       STA $20
F997    A9 FF       LDA #$FF
F999    85 48       STA $48
F99B    60          RTS

F99C    AD 07 1C    LDA $1C07
F99F    8D 05 1C    STA $1C05
F9A2    AD 00 1C    LDA $1C00
F9A5    29 10       AND #$10      write protect?
F9A7    C5 1E       CMP $1E
F9A9    85 1E       STA $1E
F9AB    F0 04       BEQ $F9B1
F9AD    A9 01       LDA #$01
F9AF    85 1C       STA $1C
F9B1    AD FE 02    LDA $02FE
F9B4    F0 15       BEQ $F9CB
F9B6    C9 02       CMP #$02
F9B8    D0 07       BNE $F9C1
F9BA    A9 00       LDA #$00
F9BC    8D FE 02    STA $02FE
F9BF    F0 0A       BEQ $F9CB
F9C1    85 4A       STA $4A
F9C3    A9 02       LDA #$02
F9C5    8D FE 02    STA $02FE
F9C8    4C 2E FA    JMP $FA2E

F9CB    A6 3E       LDX $3E
F9CD    30 07       BMI $F9D6
F9CF    A5 20       LDA $20
F9D1    A8          TAY
F9D2    C9 20       CMP #$20
F9D4    D0 03       BNE $F9D9
F9D6    4C BE FA    JMP $FABE

F9D9    C6 48       DEC $48
F9DB    D0 1D       BNE $F9FA
F9DD    98          TYA
F9DE    10 04       BPL $F9E4
F9E0    29 7F       AND #$7F
F9E2    85 20       STA $20
```

```
F9E4    29 10      AND #$10
F9E6    F0 12      BEQ $F9FA
F9E8    AD 00 1C   LDA $1C00
F9EB    29 FB      AND #$FB       drive motor on
F9ED    8D 00 1C   STA $1C00
F9F0    A9 FF      LDA #$FF
F9F2    85 3E      STA $3E
F9F4    A9 00      LDA #$00
F9F6    85 20      STA $20
F9F8    F0 DC      BEQ $F9D6
F9FA    98         TYA
F9FB    29 40      AND #$40
F9FD    D0 03      BNE $FA02
F9FF    4C BE FA   JMP $FABE

FA02    6C 62 00   JMP ($0062)

FA05    A5 4A      LDA #$4A
FA07    10 05      BPL $FA0E
FA09    49 FF      EOR #$FF
FA0B    18         CLC
FA0C    69 01      ADC #$01
FA0E    C5 64      CMP $64
FA10    B0 0A      BCS $FA1C
FA12    A9 3B      LDA #$3B
FA14    85 62      STA $62
FA16    A9 FA      LDA #$FA       pointer $62/$63 to $FA3B
FA18    85 63      STA $63
FA1A    D0 12      BNE $FA2E
FA1C    E5 5E      SBC $5E
FA1E    E5 5E      SBC $5E
FA20    85 61      STA $61
FA22    A5 5E      LDA $5E
FA24    85 60      STA $60
FA26    A9 7B      LDA #$7B
FA28    85 62      STA $62
FA2A    A9 FA      LDA #$FA       pointer $62/$63 to $FA7B
FA2C    85 63      STA $63
FA2E    A5 4A      LDA $4A        step counter for head transport
FA30    10 31      BPL $FA63
FA32    E6 4A      INC $4A        increment
FA34    AE 00 1C   LDX $1C00
FA37    CA         DEX
FA38    4C 69 FA   JMP $FA69

********************************
FA3B    A5 4A      LDA $4A        step counter for head transport
FA3D    D0 EF      BNE $FA2E      not yet zero?
FA3F    A9 4E      LDA #$4E
FA41    85 62      STA $62
FA43    A9 FA      LDA #$FA       pointer $62/$63 to $FA4E
FA45    85 63      STA $63
FA47    A9 05      LDA #$05
FA49    85 60      STA $60        counter to 5
FA4B    4C BE FA   JMP $FABE
```

```
********************************
FA4E   C6 60      DEC $60       decrement counter
FA50   D0 6C      BNE $FABE     not yet zero?
FA52   A5 20      LDA $20
FA54   29 BF      AND #$BF      erase bit 6
FA56   85 20      STA $20
FA58   A9 05      LDA #$05
FA5A   85 62      STA $62
FA5C   A9 FA      LDA #$FA      pointer $62/$63 to FA05
FA5E   85 63      STA $63
FA60   4C BE FA   JMP $FABE

********************************
FA63   C6 4A      DEC $4A       step counter for head transport
FA65   AE 00 1C   LDX $1C00
FA68   E8         INX
FA69   8A         TXA
FA6A   29 03      AND #$03
FA6C   85 4B      STA $4B
FA6E   AD 00 1C   LDA $1C00
FA71   29 FC      AND #$FC
FA73   05 4B      ORA $4B       stepper motor off
FA75   8D 00 1C   STA $1C00
FA78   4C BE FA   JMP $FABE

********************************
FA7B   38         SEC
FA7C   AD 07 1C   LDA $1C07
FA7F   E5 5F      SBC $5F
FA81   8D 05 1C   STA $1C05
FA84   C6 60      DEC $60       decrement counter
FA86   D0 0C      BNE $FA94     not yet zero?
FA88   A5 5E      LDA $5E
FA8A   85 60      STA $60
FA8C   A9 97      STA #$97
FA8E   85 62      STA $62
FA90   A9 FA      LDA #$FA      pointer $62/$63 to $FA97
FA92   85 63      STA $63
FA94   4C 2E FA   JMP $FA2E

********************************
FA97   C6 61      DEC $61
FA99   D0 F9      BNE $FA94
FA9B   A9 A5      LDA #$A5
FA9D   85 62      STA $62
FA9F   A9 FA      LDA #$FA      pointer $62/$63 to $FAA5
FAA1   85 63      STA $63
FAA3   D0 EF      BNE $FA94

********************************
FAA5   AD 07 1C   LDA $1C07
FAA8   18         CLC
FAA9   65 5F      ADC $5F
FAAB   8D 05 1C   STA $1C05
```

```
FAAE    C6 60       DEC $60       decrement counter
FAB0    D0 E2       BNE $FA94     not yet zero?
FAB2    A9 4E       LDA #$4E
FAB4    85 62       STA $62
FAB6    A9 FA       LDA #$FA      pointer $62/$63 to $FA4E
FAB8    85 63       STA $63
FABA    A9 05       LDA #$05
FABC    85 60       STA $60       counter to 5
FABE    AD 0C 1C    LDA $1C0C
FAC1    29 FD       AND #$FD      erase bit 1
FAC3    8D 0C 1C    STA $1C0C
FAC6    60          RTS

********************************  formatting
FAC7    A5 51       LDA $51       track number
FAC9    10 2A       BPL $FAF5     fomatting already in progress
FACB    A6 3D       LDX $3D       drive number
FACD    A9 60       LDA #$60      flag for head transport
FACF    95 20       STA $20,X     set
FAD1    A9 01       LDA #$01
FAD3    95 22       STA $22,X     set destination track
FAD5    85 51       STA $51       running track # for format
FAD7    A9 A4       LDA #$A4      164
FAD9    85 4A       STA $4A       step counter for head transport
FADB    AD 00 1C    LDA $1C00
FADE    29 FC       AND #$FC      stepper motor on
FAE0    8D 00 1C    STA $1C00
FAE3    A9 0A       LDA #$0A      10
FAE5    8D 20 06    STA $0620     error counter
FAE8    A9 A0       LDA #$40      $621/$622 = 4000
FAEA    8D 21 06    STA $0621     initialize track capacity
FAED    A9 0F       LDA #$0F      4000 < capacity < 2*4000 bytes
FAEF    8D 22 06    STA $0622
FAF2    4C 9C F9    JMP $F99C     back in job loop

FAF5    A0 00       LDY #$00
FAF7    D1 32       CMP ($32),Y
FAF9    F0 05       BEQ $FB00
FAFB    91 32       STA ($32),Y
FAFD    4C 9C F9    JMP $F99C     to job loop

FB00    AD 00 1C    LDA $1C00
FB03    29 10       AND #$10      write protect?
FB05    D0 05       BNE $FB0C     no
FB07    A9 08       LDA #$08
FB09    4C D3 FD    JMP $FDD3     26, 'write protect on'

FB0C    20 A3 FD    JSR $FDA3     write $FF to disk 10240 times
FB0F    20 C3 FD    JSR $FDC3     code ($621/$622) times to disk
FB12    A9 55       LDA #$55      $55
FB14    8D 01 1C    STA $1C01     to write head
FB17    20 C3 FD    JSR $FDC3     and ($621/$622) times to disk
FB1A    20 00 FE    JSR $FE00     switch to read
FB1D    20 56 F5    JSR $F556     set timer, find $FF (SYNC)
FB20    A9 40       LDA #$40
```

```
FB22   0D 0B 18   ORA $180B     timer 1 free running
FB25   8D 0B 18   STA $180B
FB28   A9 62      LDA #$62      98 cycles, about 0.1 ms
FB2A   8D 06 18   STA $1806
FB2D   A9 00      LDA #$00
FB2F   8D 07 18   STA $1807
FB32   8D 05 18   STA $1805     start timer
FB35   A0 00      LDY #$00      counter to zero
FB37   A2 00      LDX #$00
FB39   2C 00 1C   BIT $1C00     SYNC found?
FB3C   30 FB      BMI $FB39     no, wait
FB3E   2C 00 1C   BIT $1C00     SYNC found?
FB41   10 FB      BPL $FB3E     wait for SYNC
FB43   AD 04 18   LDA $1804     reset interrupt flag timer
FB46   2C 00 1C   BIT $1C00     SYNC found?
FB49   10 11      BPL $FB5C     not SYNC ($55)?
FB4B   AD 0D 18   LDA $180D     interrupt flag register
FB4E   0A         ASL A         shift timer flag
FB4F   10 F5      BPL $FB46     timer not run down yet?
FB51   E8         INX           increment counter
FB52   D0 EF      BNE $FB43
FB54   C8         INY           increment hi-byte of counter
FB55   D0 EC      BNE $FB43
FB57   A9 02      LDA #$02      overflow, then error
FB59   4C D3 FD   JMP $FDD3     20, 'read error'

FB5C   86 71      STX $71
FB5E   84 72      STY $72
FB60   A2 00      LDX #$00
FB62   A0 00      LDY #$00      counter to zero again
FB64   AD 04 18   LDA $1804     reset timer 1 interrupt flag
FB67   2C 00 1C   BIT $1C00     SYNC found?
FB6A   30 11      BMI $FB7D     yes
FB6C   AD 0D 18   LDA $180D     interrupt-flag register
FB6F   0A         ASL A         timer flag to bit 7
FB70   10 F5      BPL $FB67     no, wait until timer run down
FB72   E8         INX
FB73   D0 EF      BNE $FB64     increment counter
FB75   C8         INY
FB76   D0 EC      BNE $FB64
FB78   A9 02      LDA #$02      overflow, then error
FB7A   4C D3 FD   JMP $FDD3     20, 'read error'

FB7D   38         SEC
FB7E   8A         TXA
FB7F   E5 71      SBC $71       difference between counter
FB81   AA         TAX
FB82   85 70      STA $70
FB84   98         TYA           and value for $FF-storage
FB85   E5 72      SBC $72
FB87   A8         TAY           bring to $70/$71
FB88   85 71      STA $71
FB8A   10 0B      BPL $FB97     difference positive?
FB8C   49 FF      EOR #$FF
FB8E   A8         TAY
```

```
FB8F    8A          TXA
FB90    49 FF       EOR #$FF      calculate abs. val of difference
FB92    AA          TAX
FB93    E8          INX
FB94    D0 01       BNE $FB97
FB96    C8          INY
FB97    98          TYA
FB98    D0 04       BNE $FB9E
FB9A    E0 04       CPX #$04      difference less than 4 * 0.1 ms
FB9C    90 18       BCC $FBB6     yes
FB9E    06 70       ASL $70
FBA0    26 71       ROL $71       double difference
FBA2    18          CLC
FBA3    A5 70       LDA $70
FBA5    6D 21 06    ADC $0621
F8A8    8D 21 06    STA $0621     add to 4000
FBAB    A5 71       LDA $71
FBAD    6D 22 06    ADC $0622
FBB0    8D 22 06    STA $0622
FBB3    4C 0C FB    JMP $FB0C     repeat until diff < 4 * 0.1 ms

FBB6    A2 00       LDX #$00
FBB8    A0 00       LDY #$00      counter to zero
FBBA    B8          CLV
FBBB    AD 00 1C    LDA $1C00     SYNC?
FBBE    10 0E       BPL $FBCE     no
FBC0    50 59       BVC $FBBB     byte ready?
FBC2    B8          CLV
FBC3    E8          INX
FBC4    D0 F5       BNE $FBBB     increment counter
FBC6    C8          INY
FBC7    D0 F2       BNE $FBBB
FBC9    A9 03       LDA #$03      overflow, then error
FBCB    4C D3 FD    JMP $FDD3     21, read error

FBCE    8A          TXA
FBCF    0A          ASL A         double counter
FBD0    8D 25 06    STA $0625
FBD3    98          TYA
FBD4    2A          ROL A          and to $624/$625 as track cap.
FBD5    8D 24 06    STA $0624
FBD8    A9 BF       LDA #$BF
FBDA    2D 0B 18    AND $180B
FBDD    8D 0B 18    STA $180B
FBE0    A9 66       LDA #$66      102
FBE2    8D 26 06    STA $0626
FBE5    A6 43       LDX $43       number of sectors in this track
FBE7    A0 00       LDY #$00
FBE9    98          TYA
FBEA    18          CLC
FBEB    6D 26 06    ADC $0626
FBEE    90 01       BCC $FBF1
FBF0    C8          INY
FBF1    C8          INY
FBF2    CA          DEX
```

```
FBF3    D0 F5       BNE $FBEA      calculate # of bytes
FBF5    49 FF       EOR #$FF
FBF7    38          SEC
FBF8    69 00       ADC #$00
FBFA    18          CLC
FBFB    6D 25 06    ADC $0625
FBFE    B0 03       BCS $FC03
FC00    CE 24 06    DEC $0624
FC03    AA          TAX
FC04    98          TYA
FC05    49 FF       EOR #$FF
FC07    38          SEC
FC08    69 00       ADC #$00
FC0A    18          CLC
FC0B    6D 24 06    ADC $0624      result in A/X
FC0E    10 05       BPL $FC15
FC10    A9 04       LDA #$04
FC12    4C D3 FD    JMP $FDD3      22, 'read error'

FC15    A8          TAY
FC16    8A          TXA
FC17    A2 00       LDX #$00
FC19    38          SEC            total divided by number
FC1A    E5 43       SBC $43        of sectors ($43)
FC1C    B0 03       BCS $FC21
FC1E    88          DEY
FC1F    30 03       BMI $FC24
FC21    E8          INX
FC22    D0 F5       BNE $FC19
FC24    8E 26 06    STX $0626      compare no. of bytes per interval
FC27    E0 04       CPX #$04       with minimum value
FC29    B0 05       BCS $FC30      ok
FC2B    A9 05       LDA #$05
FC2D    4C D3 FD    JMP $FDD3      23, 'read error'
FC30    18          CLC            remainder of division
FC31    65 43       ADC $43        plus number of sectors
FC33    8D 27 06    STA $0627      save
FC36    A9 00       LDA #$00
FC38    8D 28 06    STA $0628      counter for sectors
FC3B    A0 00       LDY #$00       counter lo
FC3D    A6 3D       LDX $3D        drive number
FC3F    A5 39       LDA $39        constant 8, marker for heacer
FC41    99 00 03    STA $0300,Y    in buffer
FC44    C8          INY
FC45    C8          INY
FC46    AD 28 06    LDA $0628      sector number
FC49    99 00 03    STA $0300,Y    in buffer
FC4C    C8          INY
FC4D    A5 51       LDA $51        track number
FC4F    99 00 03    STA $0300,Y    in buffer
FC52    C8          INY
FC53    B5 13       LDA $13,X      ID 2
FC55    99 00 03    STA $0300,Y    in buffer
FC58    C8          INY
FC59    B5 12       LDA $12,X      ID 1
```

```
FC5B    99 00 03   STA $0300,Y    in buffer
FC5E    C8         INY
FC5F    A9 0F      LDA #$0F       15
FC61    99 00 03   STA $0300,Y    in buffer
FC64    C8         INY
FC65    99 00 03   STA $0300,Y    15 in buffer
FC68    C8         INY
FC69    A9 00      LDA #$00
FC6B    59 FA 02   EOR $02FA,Y
FC6E    59 FB 02   EOR $02FB,Y
FC71    59 FC 02   EOR $02FC,Y    generate checksum
FC74    59 FD 02   EOR $02FD,Y
FC77    99 F9 02   STA $02F9,Y
FC7A    EE 28 06   INC $0628      increment counter
FC7D    AD 28 06   LDA $0628      counter
FC80    C5 43      CMP $43        compare with no. of sectors
FC82    90 BB      BCC $FC3F      smaller, then continue
FC84    98         TYA
FC85    48         PHA
FC86    E8         INX
FC87    8A         TXA
FC88    9D 00 05   STA $0500,X
FC8B    E8         INX
FC8C    D0 FA      BNE $FC88
FC8E    A9 03      LDA #$03       buffer pointer to $300
FC90    85 31      STA $31
FC92    20 30 FE   JSR $FE30
FC95    68         PLA
FC96    A8         TAY
FC97    88         DEY
FC98    20 E5 FD   JSR $FDE5      copy buffer data
FC9B    20 F5 FD   JSR $FDF5      copy data in buffer
FC9E    A9 05      LDA #$05
FCA0    85 31      STA $31        buffer pointer to $500
FCA2    20 E9 F5   JSR $F5E9      calculate parity for data buffer
FCA5    85 3A      STA $3A        and save
FCA7    20 8F F7   JSR $F78F
FCAA    A9 00      LDA #$00
FCAC    85 32      STA $32
FCAE    20 0E FE   JSR $FE0E
FCB1    A9 FF      LDA #$FF
FCB3    8D 01 1C   STA $1C01      to write head
FCB6    A2 05      LDX #$05       write $FF 5 times
FCB8    50 FE      BVC $FCB8      byte ready
FCBA    B8         CLV
FCBB    CA         DEX
FCBC    D0 FA      BNE $FCB8
FCBE    A2 0A      LDX #$0A       10 times
FCC0    A4 32      LDY $32        buffer pointer
FCC2    50 FE      BVC $FCC2      byte ready?
FCC4    B8         CLV
FCC5    B9 00 03   LDA $0300,Y    data from buffer
FCC8    8D 01 1C   STA $1C01      write
FCCB    C8         INY
FCCC    CA         DEX            10 data written?
```

```
FCCD   D0 F3      BNE $FCC2
FCCF   A2 09      LDX #$09       9 times
FCD1   50 FE      BVC $FCD1      byte ready?
FCD3   B8         CLV
FCD4   A9 55      LDA #$55       $55
FCD6   8D 01 1C   STA $1C01      write
FCD9   CA         DEX
FCDA   D0 F5      BNE $FCD1      9 times?
FCDC   A9 FF      LDA #$FF       $FF
FCDE   A2 05      LDX #$05       5 times
FCE0   50 FE      BVC $FCE0      byte ready?
FCE2   B8         CLV
FCE3   8D 01 1C   STA $1C01      to write head
FCE6   CA         DEX
FCE7   D0 F7      BNE $FCE0
FCE9   A2 BB      LDX #$BB
FCEB   50 FE      BVC $FCEB
FCED   B8         CLV
FCEE   BD 00 01   LDA $0100,X    area $1BB to $1FF
FCF1   8D 01 1C   STA $1C01      save
FCF4   E8         INX
FCF5   D0 F4      BNE $FCEB
FCF7   A0 00      LDY #$00
FCF9   50 FE      BVC $FCF9      byte ready?
FCFB   B8         CLV
FCFC   B1 30      LDA ($30),Y    256 bytes of data
FCFE   8D 01 1C   STA $1C01      write byte to disk
FD01   C8         INY
FD02   D0 F5      BNE $FCF9
FD04   A9 55      LDA #$55       $55
FD06   AE 26 06   LDX $0626      ($626) times
FD09   50 FE      BVC $FD09
FD0B   B8         CLV
FD0C   8D 01 1C   STA $1C01      write
FD0F   CA         DEX
FD10   D0 F7      BNE $FD09
FD12   A5 32      LDA $32
FD14   18         CLC
FD15   69 0A      ADC #$0A       plus 10
FD17   85 32      STA $32
FD19   CE 28 06   DEC $0628      decrement sector number
FD1C   D0 93      BNE $FCB1
FD1E   50 FE      BVC $FD1E      byte ready?
FD20   B8         CLV
FD21   50 FE      BVC $FD21      byte ready?
FD23   B8         CLV
FD24   20 00 FE   JSR $FE00      switch to reading
FD27   A9 C8      LDA #$C8       200
FD29   8D 23 06   STA $0623
FD2C   A9 00      LDA #$00
FD2E   85 30      STA $30
FD30   A9 03      LDA #$03       buffer pointer to $200
FD32   85 31      STA $31
FD34   A5 43      LDA $43        number of sectors per track
FD36   8D 28 06   STA $0628
```

```
FD39    20 56 F5    JSR $F556      wait for SYNC
FD3C    A2 0A       LDX #$0A       10 data
FD3E    A0 00       LDY #$00
FD40    50 FE       BVC $FD40      byte ready?
FD42    B8          CLV
FD43    AD 01 1C    LDA $1C01      read byte
FD46    D1 30       CMP ($30),Y    compare with data in buffer
FD48    D0 0E       BNE $FD58      not equal, error
FD4A    C8          INY
FD4B    CA          DEX
FD4C    D0 F2       BNE $FD40
FD4E    18          CLC
FD4F    A5 30       LDA $30
FD51    69 0A       ADC #$0A       increment pointer by 10
FD53    85 30       STA $30
FD55    4C 62 FD    JMP $FD62

FD58    CE 23 06    DEC $0623      decrement counter for attempts
FD5B    D0 CF       BNE $FD2C      not yet zero?
FD5D    A9 06       LDA #$06       else error
FD5F    4C D3 FD    JMP $FDD3      24, 'read error'

FD62    20 56 F5    JSR $F556      wait for SYNC
FD65    A0 BB       LDY #$BB
FD67    50 FE       BVC $FD67      byte ready?
FD69    B8          CLV
FD6A    AD 01 1C    LDA $1C01      read byte
FD6D    D9 00 01    CMP $0100,Y    compare with buffer contents
FD70    D0 E6       BNE $FD58      not equal, error
FD72    C8          INY
FD73    D0 F2       BNE $FD67      next byte
FD75    A2 FC       LDX #$FC
FD77    50 FE       BVC $FD77      byte ready?
FD79    B8          CLV
FD7A    AD 01 1C    LDA $1C01      read byte
FD7D    D9 00 05    CMP $0500,Y    compare with buffer contents
FD80    D0 D6       BNE $FD58      not equal, then error
FD82    C8          INY
FD83    CA          DEX            next byte
FD84    D0 F1       BNE $FD77
FD86    CE 28 06    DEC $0628      decrement sector counter
FD89    D0 AE       BNE $FD39      not yet zero?
FD8B    E6 51       INC $51        increment track number
FD8D    A5 51       LDA $51
FD8F    C9 24       CMP #$24       compare with 36, highest trk# +1
FD91    B0 03       BCS $FD96      greater, then formatting done
FD93    4C 9C F9    JMP $F99C      continue

FD96    A9 FF       LDA #$FF
FD98    85 51       STA $51        track number to $FF
FD9A    A9 00       LDA #$00
FD9C    85 50       STA $50
FD9E    A9 01       LDA #$01
FDA0    4C 69 F9    JMP $F969      ok
```

```
********************************   write $FF 10240 times
FDA3    AD 0C 1C    LDA $1C0C
FDA6    29 1F       AND #$1F        switch PCR to writing
FDA8    09 C0       ORA #$C0
FDAA    8D 0C 1C    STA $1C0C
FDAD    A9 FF       LDA #$FF
FDAF    8D 03 1C    STA $1C03       port A(read/write head) to output
FDB2    8D 01 1C    STA $1C01       write $FF to disk
FDB5    A2 28       LDX #$28        40
FDB7    A0 00       LDY #$00
FDB9    50 FE       BVC $FDB9       byte ready?
FDBB    B8          CLV
FDBC    88          DEY
FDBD    D0 FA       BNE $FD89
FDBF    CA          DEX
FDC0    D0 F7       BNE $FD89
FDC2    60          RTS

********************************   read/write ($621/$622) times
FDC3    AE 21 06    LDX $0621
FDC6    AC 22 06    LDY $0622
FDC9    50 FE       BVC $FDC9       byte ready?
FDCB    B8          CLV
FDCC    CA          DEX
FDCD    D0 FA       BNE $FDC9
FDCF    88          DEY
FDD0    10 F7       BPL $FDC9
FDD2    60          RTS

********************************   attempt counter for formatting
FDD3    CE 20 06    DEC $0620       decrement number of attempts
FDD6    F0 03       BEQ $FDDB       zero, then error
FDD8    4C 9C F9    JMP $F99C       continue

FDDB    A0 FF       LDY #$FF
FDDD    84 51       STY $51         flag for end of formatting
FDDF    C8          INY
FDE0    84 50       STY $50
FDE2    4C 69 F9    JMP $F969       error termination

********************************
FDE5    B9 00 03    LDA $0300,Y
FDE8    99 45 03    STA $0345,Y
FDEB    88          DEY             copy buffer contents
FDEC    D0 F7       BNE $FDE5
FDEE    AD 00 03    LDA $0300
FDF1    8D 45 03    STA $0345
FDF4    60          RTS

********************************
FDF5    A0 44       LDY #$44
FDF7    B9 BB 01    LDA $01BB,Y     $1BB to $1FF
FDFA    91 30       STA ($30),Y     write in buffer $30/$31
FDFC    88          DEY
FDFD    10 F8       BPL $FDF7
```

```
FDFF    60          RTS

******************************  switch to reading
FE00    AD 0C 1C    LDA $1C0C
FE03    09 E0       ORA #$E0      switch PCR to reading
FE05    8D 0C 1C    STA $1C0C
FE08    A9 00       LDA #$00
FE0A    8D 03 1C    STA $1C03     port A to input
FE0D    60          RTS

******************************  write $55 10240 times
FE0E    AD 0C 1C    LDA $1C0C
FE11    29 1F       AND #$1F
FE13    09 C0       ORA #$C0      switch PCR to writing
FE15    8D 0C 1C    STA $1C0C
FE18    A9 FF       LDA #$FF
FE1A    8D 03 1C    STA $1C03     port A to output (write head)
FE1D    A9 55       LDA #$55      %01010101
FE1F    8D 01 1C    STA $1C01     to port A (write head)
FE22    A2 28       LDX #$28
FE24    A0 00       LDY #$00
FE26    50 FE       BVC $FE26     byte ready for write electronics
FE28    B8          CLV
FE29    88          DEY
FE2A    D0 FA       BNE $FE26     10240 times
FE2C    CA          DEX
FE2D    D0 F7       BNE $FE26
FE2F    60          RTS

******************************
FE30    A9 00       LDA #$00
FE32    85 30       STA $30
FE34    85 2E       STA $2E
FE36    85 36       STA $36
FE38    A9 BB       LDA #$BB
FE3A    85 34       STA $34
FE3C    A5 31       LDA $31
FE3E    85 2F       STA $2F
FE40    A9 01       LDA #$01
FE42    85 31       STA $31
FE44    A4 36       LDY $36
FE46    B1 2E       LDA ($2E),Y
FE48    85 52       STA $52
FE4A    C8          INY
FE4B    B1 2E       LDA ($2E),Y
FE4D    85 53       STA $53
FE4F    C8          INY
FE50    B1 2E       LDA ($2E),Y
FE52    85 54       STA $54
FE54    C8          INY
FE55    B1 2E       LDA ($2E),Y
FE57    85 55       STA $55
FE59    C8          INY
FE5A    F0 08       BEQ $FE64
FE5C    84 36       STY $36
```

```
FE5E    02 D0 F6    JSR $F6D0
FE61    4C 44 FE

FE64    4C D0 F6    JMP $F6D0

*********************************   interrupt routine
FE67    48          PHA
FE68    8A          TXA
FE69    48          PHA               save registera
FE6A    98          TYA
FE6B    48          PHA
FE6C    AD 0D 18    LDA $180D         interrupt from serial bus
FE6F    29 02       AND #$02
FE71    F0 03       BEQ $FE76         no
FE73    20 53 E8    JSR $E853         serve serial bus
FE76    AD 0D 1C    LDA $1C0D         interrupt from timer 1?
FE79    0A          ASL A
FE7A    10 03       BPL $FE7F         no
FE7C    20 B0 F2    JSR $F2B0         IRQ routine for disk controller
FE7F    68          PLA
FE80    A8          TAY
FE81    68          PLA               get register back
FE82    AA          TAX
FE83    68          PLA
FE84    40          RTI

*********************************   constants for disk format
FE85 12                               18, track for BAM and directory
FE86 04                               start of BAM at position 4
FE87 04                               4 bytes in BAM for each track
FE88 90                               $90 = 144, end of BAM, disk name

*********************************   table of command words
FE89 56 49 44 4D 42 55                'V', 'I', 'D', 'M', 'B', 'U'
FE8F 50 26 43 52 53 4E                'P', '&', 'C', 'R', 'S', 'N'

*********************************   lo-bytes of command addresses
FE95 84 05 C1 F8 1B 5C
FE9F 07 A3 F0 88 23 0D

*********************************   hi-bytes of command addresses
FEA1 ED D0 C8 CA CC CB
FEA7 E2 E7 C8 CA C8 FE

*********************************
FEAD 51 DD 1C 9E 1C                   bytes for syntax check

*********************************   file control methods
FEB2 52 57 41 4D                      'R', 'W', 'A', 'M'

*********************************   file types
FEB6 44 53 50 55 4C                   'D', 'S', 'P', 'U', 'L'

*********************************   names of file types
FEBB 44 53 50 55 52              1st letters 'D', 'S', 'P', 'U', 'R'
```

```
FEC0 45 45 52 53 45        2nd letters 'E', 'E', 'R', 'S', 'E'
FEC5 4C 51 47 52 4C        3rd letters 'L', 'Q', 'G', 'R', 'L'

********************************
FECA 08 00 00

********************************
FECD 3F 7F BF FF                     masks for bit command

********************************     number of sectors per track
FED1 11 12 13 15                     17, 18, 19, 21

********************************     contants for disk format
FED5 4A                              'A' marker for 1541 format
FED6 04                              4 track numbers
FED7 24                              36, highest track number + 1
FED8 1F 19 12                        31, 25, 18 tracks with change of
                                     number of sectors

********************************
FEDB 01 FF FF 01 00                  control bytes for head position

********************************     addresses of buffers
FEE0 03 04 05 06 07                  high bytes

********************************
FEE5 07 0E

********************************     for UI command
FEE7   6C 65 00   JMP ($0065)

********************************     for diagnostic routine
FEEA   8D 00 1C   STA $1C00          turn LED on
FEED   8D 02 1C   STA $1C02          port to output
FEF0   4C 7D EA   JMP $EA7D          back to diagnostic routine

********************************     delay loop for serial bus
FEF3   8A         TXA
FEF4   A2 05      LDX #$05
FEF6   CA         DEX                about 40 microseconds
FEF7   D0 FD      BNE $FEF6
FEF9   AA         TAX
FEFA   60         RTS

********************************     data output to serial bus
FEFB   20 AE E9   JSR $E9AE          CLOCK OUT hi
FEFE   4C 9C E9   JMP $E99C          DATA OUT lo

********************************     UI vector
FF01   AD 02 02   LDA $0202
FF04   C9 2D      CMP #$2D           '-'
FF06   F0 05      BEQ $FF0D
FF08   38         SEC
FF09   E9 2B      SBC #$2B           '+'
FF0B   D0 DA      BNE $FEE7          indirect jump over ($65)
```

```
FF0D   85 23      STA $23
FF0F   60         RTS

******************************
FF10 AA ...
FFE1 ... AA

******************************
FFE2 52 53 52 AA
FFE6 C6 C8 8F F9

******************************  USER vectors
FFEA 5F CD                      UA, U1, $CD5F
FFEC 97 CD                      UB, U2, $CD97
FFEE 00 05                      UC, U3, $0500
FFF0 03 05                      UD, U4, $0503
FFF2 06 05                      UE, U5, $0506
FFF4 09 05                      UF, U6, $0509
FFF6 0C 05                      UG, U7, $050C
FFF8 0F 05                      UH, U8, $050F
FFFA 01 FF                      UI, U9, $FF01
                            (NMI vector not used)

******************************  hardware vectors
FFFC 0A EA                      $EAA0   RESET and UJ (U:) vector
FFFE 67 FE                      $FE67   IRQ vector
```

APPENDICE

INPUT DI STRINGHE DI LUNGHEZZA PRESTABILITA DA DISCO

La lettura di dati dal disco con un comando INPUT# ha il grande svantaggio che solo gruppi di dati inferiori a 88 caratteri possono essere letti e cio' perche il buffer di input del computer e' di dimensioni limitate.
In oltre non tutti i caratteri possono essere letti con un comando di INPUT. Infatti se in un record ci sono caratteri virgola il basic li interpreta come cartteri di separazione e quindi la parte rimanente dei dati in ingresso viene assegnata alla successiva variabile.
L' alternativa sarebbe quella di utilizzare un comando di GET# ripetuto per N caratteri ma il risultato sarebbe un esecuzione troppo lenta.
Per evitare questi problemi si puo' usare una piccola routine in linguaggio macchina.
Cambieremo il comando INPUT# in maniera tale che si possa dichiarare il numero di caratteri che devono essere letti e per distinguerlo dal normale comando INPUT# lo chiameremo **INPUT***.
La sintassi del comando sara' la seguente:

```
  INPUT* LFN,LEN,VAR
```

In cui LFN sara' il numero di file logico preventivamente aperto, LEN il numero di caratteri che devono essere letti e VAR la variabile stringa entro cui i caratteri verranno letti.
Un programma potrebbe essere il seguente:

```
100 OPEN 2,8,2,"FILE"
110 INPUT*2,100,A$
```

Questa routine legge una stringa di 100 caratteri da un file aperto e la mette in A$.

Di seguito riportiamo un listato in assembler che risiede nel buffer di cassetta.

```
63000 REM******************
63010 REM   INPUT
63020 REM******************
63200 FORI=828TO922
63205 READX:POKEI,X:S=S+X:NEXT
63210 DATA169,71,160,3,141,8,3,140,9,3,96,32
63215 DATA115,0,201,133,240,6,32,121,0,76,231,167
63220 DATA32,115,0,201,172,240,6,32,191,171,76,174
63225 DATA167,32,155,183,32,30,225,32,253,174,32,158
63230 DATA183,138,72,32,253,174,32,139,176,133,73,132
63235 DATA74,32,163,182,104,32,117,180,160,2,185,97
63240 DATA0,145,73,136,16,248,200,32,18,225,145,98
63245 DATA200,196,97,208,246,32,204,255,76,174,167
63247 IFS<>11096THENPRINT"ERRORE": END
63250 SYS828
```

```
 0  033C  A947     LDA  #$47
 1  033E  A003     LDY  #$03
 2  0340  8D0803   STA  $0308
 3  0343  8C0903   STY  $0309
 4  0346  60       RTS
 5  0347  207300   JSR  $0073
 6  034A  C985     CMP  #$85
 7  034C  F006     BEQ  $0354
 8  034E  207900   JSR  $0079
 9  0351  4CE7A7   JMP  $A7E7
10  0354  207300   JSR  $0073
11  0357  C9AC     CMP  #$AC
12  0359  F006     BEQ  $0361
13  035B  20BFAB   JSR  $ABBF
14  035E  4CAEA7   JMP  $A7AE
15  0361  209BB7   JSR  $B79B
16  0364  201EE1   JSR  $E11E
17  0367  20FDAE   JSR  $AEFD
18  036A  209EB7   JSR  $B79E
19  036D  8A       TXA
20  036E  48       PHA
21  036F  20FDAE   JSR  $AEFD
22  0372  208BB0   JSR  $B08B
23  0375  8549     STA  $49
24  0377  844A     STY  $4A
25  0379  20A3B6   JSR  $B6A3
26  037C  68       PLA
27  037D  2075B4   JSR  $B475
28  0380  A002     LDY  #$02
29  0382  B96100   LDA  $0061,Y
30  0385  9149     STA  ($49),Y
31  0387  88       DEY
32  0388  10F8     BPL  $0382
33  038A  C8       INY
34  038B  2012E1   JSR  $E112
35  038E  9162     STA  ($62),Y
36  0390  C8       INY
37  0391  C461     CPY  $61
38  0393  D0F6     BNE  $038B
39  0395  20CCFF   JSR  $FFCC
40  0398  4CAEA7   JMP  $A7AE
```

SPOOLING STAMPA DIRETTA DA DISCO

Se avete una stampante connessa al vostro computer oltre alla unita' a dischi potete usare una caratteristica del bus seriale.
E' possibile inviare FILES direttamente da disco alla stampante senza la necessita' che essi vengano trasferiti byte per byte dal disco al computer e da qui alla stampante.
Per esempio se avete un testo memorizzato come file sequenziale e lo volete stampare potete utilizzare il seguente programma:

```
100 OPEN 1,4
110 OPEN 2,8,2,"0:TEST"
120 GET#2,A$:IF ST= 64 THEN140
130 PRINT#1,A$:GOTO120
140 CLOSE1 : CLOSE2
150 END
```

I caratteri sono inviati dal disco alla stampante fino a quando non venga trovata la fine del File. Questa sarebbe la normale procedura in basic. Con la nostra routine invece sara' sufficiente digitare:

```
SYS 828,"NOME FILE"
```

Il grande vantaggio sta nel fatto di non impegnare l'unita' centrale durante la fase di stampa.

NOTA

Con stampanti non COMMODORE il programma puo' dare degli inconvenienti.

```
0 PRINT"        [CLR HOME]"
1 PRINT " ********************************
2 PRINT"         SPOOLING * MPS 801
3 PRINT" ********************************
4 PRINT"  FUNZIONA SOLO CON MPS 801
5 PRINT" ********************************
6 PRINT"   PER IL FUNZIONAMENTO
7 PRINT"   OPEN4,4:SYS828,"CHR$(34)"NOME DEL FILE"CHR$(34)
8 PRINT"   PER CHIUDERE IL FILE AD
9 PRINT"   OPERAZIONE ESEGUITA
10 PRINT"   SYS 828
12 FORI=828TO901
20 READX:POKEI,X:S=S+X:NEXT
100 DATA 32,121,  0,240, 51, 32,231,255, 32, 84,226
110 DATA166,183,240, 56,169,  2,162,  8,160,  2, 32
120 DATA186,255, 32,192,255,169,  4, 32,177,255, 32
130 DATA190,237,162,  2, 32,198,255, 32,190,237, 32
140 DATA133,238, 32,151,238,169,  0,133,153,133,152
150 DATA 96,169,  1,133,152, 32,174,255, 32,171,255
160 DATA169,  2, 76,195,255, 76,  8,175
200 IFS<>9598THENPRINT"ERRORE IN DATA !!!":END
210 PRINT"OK"
```

DISK MONITOR

Vi presentiamo qui una delle migliori utility per il drive 1541, che vi consentira' di visualizzare, caricare modificare e salvare un qualsiasi blocco su dischetto.
Per ovvie ragioni di velocita' il programma e' stato scritto interamente in linguaggio macchina.
E' importante notare che i parametri utilizzati nei comandi DEVONO essere dati in ESADECIMALE.
I comadi implementati sono i seguenti:

>R TT SS : Lettura di un blocco

>W TT SS : Scrittura di un blocco

>M IN FN : Visualizzazione di un blocco da IN a FN

>: : Edit di un blocco

> @ : Visualizzazione messaggio di errore

>@(comandi) : Invio di comandi al disco

>X : Ritorno al BASIC

TT = Traccia in ESADECIMALE

SS = Settore in ESADECIMALE

NOTA

Se dopo essere tornati in ambito BASIC con il comando X desiderate utilizzare ancora il DISK MONITOR non e' necessario che questo venga ricaricato. Sara' sufficiente digitare SYS 49152.

```
1 REM**** EVM COMPUTERS********
2 REM**** DISK MONITOR ********
10 FORI=49152TO49867
20 READX:POKEI,X:S=S+X:NEXT
30 DATA162,  0,169,133,194, 32,210,255,232,224, 16,208
40 DATA245,162, 13,169, 62, 32,235,192,169,  0,141,  1
50 DATA  2, 32, 51,193,201, 62,240,249,201, 32,240,245
60 DATA162,  5,221,106,192,208, 12,142,  0,  2,189,112
70 DATA192, 72,189,118,192, 72, 96,202, 16,236, 76, 13
80 DATA192,133,151, 32, 98,192,185,224,194, 32,220,192
90 DATA200,208,  3,238,  1,  2,198,151,208,237, 96, 32
100 DATA254,192,144,  3,153,224,194,200,198,151, 96, 32
110 DATA 98,192,169, 32, 44,169, 13, 76,210,255, 58, 87
120 DATA 82, 77, 64, 88,192,193,193,192,193,227,192,144
130 DATA144,123, 62,122,160,  0,140,  3,  2,136,140,  4
140 DATA  2, 32,207,255,201, 13,240, 23, 32,254,192,144
150 DATA 18,141,  3,  2, 32,207,255,201, 13,240,  8, 32
160 DATA254,192,144,  3,141,  4,  2,172,  3,  2, 32,198
170 DATA194, 32,214,194,152, 32,220,192, 32, 98,192,169
180 DATA  8, 32, 61,192, 32,151,194, 76,166,192, 76, 13
190 DATA192, 32,254,192,144,248,168,169,  8,133,151, 32
200 DATA 51,193, 32, 51,193, 32, 83,192,208,248, 32,151
210 DATA194, 76, 13,192, 72, 74, 74, 74, 74, 32,244,192
220 DATA170,104, 41, 15, 32,244,192, 72,138, 32,210,255
230 DATA104, 76,210,255, 24,105,246,144,  2,105,  6,105
240 DATA 58, 96,169,  0,141,  2,  2, 32, 51,193,201, 32
250 DATA208,  9, 32, 51,193,201, 32,208, 15, 24, 96, 32
260 DATA 40,193, 10, 10, 10, 10,141,  2,  2, 32, 51,193
270 DATA 32, 40,193, 13,  2,  2, 56, 96,201, 58,  8, 41
280 DATA 15, 40,144,  2,105,  8, 96, 32,207,255,201, 13
290 DATA208,248,104,104, 76, 13,192, 32,207,255,201, 13
300 DATA208, 39,169,  0,133,144, 32,101,192,169,  8,133
310 DATA186, 32,180,255,169,111,133,185, 32,150,255, 32
320 DATA165,255, 36,144,112,  5, 32,210,255,208,244, 32
330 DATA171,255, 76, 13,192,201, 36,240, 29, 72,169,  8
```

```
340 DATA133,166, 32,177,255,169,111,133,165, 32,147,255
350 DATA104, 32,168,255, 32,207,255,201, 13,208,246, 32
360 DATA174,255, 76, 13,192, 32, 51,193, 32,254,192,144
370 DATA245,141, 39,194, 32, 51,193,32 ,254,192,144,234
380 DATA141, 42,194, 32, 73,194,173,  0,  2,201,  1,240
390 DATA 30,169, 49, 32,237,193,162, 13, 32,198,255,162
400 DATA  0, 32,207,255,157,224,194,232,208,247, 32,204
410 DATA255, 32,110,194, 76, 13,192, 32, 44,194,162, 13
420 DATA 32,201,255,162,  0,169,224,194, 32,210,255,232
430 DATA208,247, 32,204,255,169, 50, 32,237,193, 76,201
440 DATA193,141, 32,194,162, 15,173, 39,194, 32,120,194
450 DATA142, 39,194,141, 40,194,173, 42,194, 32,120,194
460 DATA142, 42,194,141, 43,194,162, 15, 32,201,255,162
470 DATA  0,169, 31,194, 32,210,255,232,224, 13,208,245
480 DATA 76,204,255, 85, 49, 58, 49, 51, 32, 48, 32,  0
490 DATA  0, 32,  0,  0,162, 15, 32,201,255,162,  0,169
500 DATA 65,194, 32,210,255,232,224,  8,208,245, 76,204
510 DATA255, 66, 45, 80, 32, 49, 51, 32, 48,169, 15,168
520 DATA162,  8, 32,186,255,169,  0, 32,189,255, 32,192
530 DATA255,169, 13,168,162,  8, 32,186,255,169,  1,162
540 DATA109,160,194, 32,189,255, 76,192,255, 35,169,13
550 DATA 32,195,255,169, 15, 76,195,255,162, 48, 56,233
560 DATA 10,144,  3,232,176,249,105, 58, 96, 13, 68, 73
570 DATA 83, 75, 45, 77, 79, 78, 73, 84, 79, 82, 32, 66
580 DATA 49, 46, 48,152, 56,233,  8,168, 32, 98,192,169
590 DATA 18, 32,210,255,162,  8,165,224,194, 41,127,201
600 DATA 32,176,  4,169, 46,208,  3,185,224,194, 32,210
610 DATA255,169,  0,133,212,200,202,208,229,169,146, 76
620 DATA210,255,173,  1,  2,208,  6,204,  4,  2,176,  1
630 DATA96 ,104,104,76 ,13 ,192, 32,101,192,169, 58,162
640 DATA62 ,76 ,235,192
650 IFS <> 90444THENPRINT"ERRORE":END
660 SYS49152
```

```
 0  C000  A200    LDX  #$00
 1  C002  BD55C2  LDA  $C255,X
 2  C005  20D2FF  JSR  $FFD2
 3  C008  E8      INX
 4  C009  E012    CPX  #$12
 5  C00B  D0F5    BNE  $C002
 6  C00D  A20D    LDX  #$0D
 7  C00F  A93E    LDA  #$3E
 8  C011  20EBC0  JSR  $C0EB
 9  C014  A900    LDA  #$00
10  C016  8D0102  STA  $0201
11  C019  2033C1  JSR  $C133
12  C01C  C93E    CMP  #$3E
13  C01E  F0F9    BEQ  $C019
14  C020  C920    CMP  #$20
15  C022  F0F5    BEQ  $C019
16  C024  A205    LDX  #$05
17  C026  DD6AC0  CMP  $C06A,X
18  C029  D00C    BNE  $C037
19  C02B  8E0002  STX  $0200
20  C02E  BD70C0  LDA  $C070,X
21  C031  48      PHA
22  C032  BD76C0  LDA  $C076,X
23  C035  48      PHA
24  C036  60      RTS
25  C037  CA      DEX
26  C038  10EC    BPL  $C026
27  C03A  4C0DC0  JMP  $C00D
28  C03D  8597    STA  $97
29  C03F  2062C0  JSR  $C062
30  C042  B9E0C2  LDA  $C2E0,Y
31  C045  20DCC0  JSR  $C0DC
32  C048  C8      INY
33  C049  D003    BNE  $C04E
34  C04B  EE0102  INC  $0201
35  C04E  C697    DEC  $97
36  C050  D0ED    BNE  $C03F
37  C052  60      RTS
38  C053  20FEC0  JSR  $C0FE
39  C056  9003    BCC  $C05B
40  C058  99E0C2  STA  $C2E0,Y
41  C05B  C8      INY
42  C05C  C697    DEC  $97
43  C05E  60      RTS
44  C05F  2062C0  JSR  $C062
45  C062  A920    LDA  #$20
46  C064  2CA90D  BIT  $0DA9
47  C067  4CD2FF  JMP  $FFD2
48  C06A  3A      BYT  $3A
49  C06B  57      BYT  $57
50  C06C  52      BYT  $52
51  C06D  4D4058  EOR  $5840
52  C070  C0C1    CPY  #$C1
53  C072  C1C0    CMP  ($C0,X)
54  C074  C1E3    CMP  ($E3,X)
55  C076  C090    CPY  #$90
56  C078  907B    BCC  $C0F5
```

```
 57  C07A  3E7AA0    ROL  $A07A,X
 58  C07D  00        BRK
 59  C07E  8C0302    STY  $0203
 60  C081  88        DEY
 61  C082  8C0402    STY  $0204
 62  C085  20CFFF    JSR  $FFCF
 63  C088  C90D      CMP  #$0D
 64  C08A  F017      BEQ  $C0A3
 65  C08C  20FEC0    JSR  $C0FE
 66  C08F  9012      BCC  $C0A3
 67  C091  8D0302    STA  $0203
 68  C094  20CFFF    JSR  $FFCF
 69  C097  C90D      CMP  #$0D
 70  C099  F008      BEQ  $C0A3
 71  C09B  20FEC0    JSR  $C0FE
 72  C09E  9003      BCC  $C0A3
 73  C0A0  8D0402    STA  $0204
 74  C0A3  AC0302    LDY  $0203
 75  C0A6  20C6C2    JSR  $C2C6
 76  C0A9  20D6C2    JSR  $C2D6
 77  C0AC  98        TYA
 78  C0AD  20DCC0    JSR  $C0DC
 79  C0B0  2062C0    JSR  $C062
 80  C0B3  A908      LDA  #$08
 81  C0B5  203DC0    JSR  $C03D
 82  C0B8  2097C2    JSR  $C297
 83  C0BB  4CA6C0    JMP  $C0A6
 84  C0BE  4C0DC0    JMP  $C00D
 85  C0C1  20FEC0    JSR  $C0FE
 86  C0C4  90F8      BCC  $C0BE
 87  C0C6  A8        TAY
 88  C0C7  A908      LDA  #$08
 89  C0C9  8597      STA  $97
 90  C0CB  2033C1    JSR  $C133
 91  C0CE  2033C1    JSR  $C133
 92  C0D1  2053C0    JSR  $C053
 93  C0D4  D0F8      BNE  $C0CE
 94  C0D6  2097C2    JSR  $C297
 95  C0D9  4C0DC0    JMP  $C00D
 96  C0DC  48        PHA
 97  C0DD  4A        LSR  A
 98  C0DE  4A        LSR  A
 99  C0DF  4A        LSR  A
100  C0E0  4A        LSR  A
101  C0E1  20F4C0    JSR  $C0F4
102  C0E4  AA        TAX
103  C0E5  68        PLA
104  C0E6  290F      AND  #$0F
105  C0E8  20F4C0    JSR  $C0F4
106  C0EB  48        PHA
107  C0EC  8A        TXA
108  C0ED  20D2FF    JSR  $FFD2
109  C0F0  68        PLA
110  C0F1  4CD2FF    JMP  $FFD2
111  C0F4  18        CLC
112  C0F5  69F6      ADC  #$F6
113  C0F7  9002      BCC  $C0FB
```

```
114  C0F9  6906     ADC  #$06
115  C0FB  693A     ADC  #$3A
116  C0FD  60       RTS
117  C0FE  A900     LDA  #$00
118  C100  8D0202   STA  $0202
119  C103  2033C1   JSR  $C133
120  C106  C920     CMP  #$20
121  C108  D009     BNE  $C113
122  C10A  2033C1   JSR  $C133
123  C10D  C920     CMP  #$20
124  C10F  D00F     BNE  $C120
125  C111  18       CLC
126  C112  60       RTS
127  C113  2028C1   JSR  $C128
128  C116  0A       ASL  A
129  C117  0A       ASL  A
130  C118  0A       ASL  A
131  C119  0A       ASL  A
132  C11A  8D0202   STA  $0202
133  C11D  2033C1   JSR  $C133
134  C120  2028C1   JSR  $C128
135  C123  0D0202   ORA  $0202
136  C126  38       SEC
137  C127  60       RTS
138  C128  C93A     CMP  #$3A
139  C12A  08       PHP
140  C12B  290F     AND  #$0F
141  C12D  28       PLP
142  C12E  9002     BCC  $C132
143  C130  6908     ADC  #$08
144  C132  60       RTS
145  C133  20CFFF   JSR  $FFCF
146  C136  C90D     CMP  #$0D
147  C138  D0F8     BNE  $C132
148  C13A  68       PLA
149  C13B  68       PLA
150  C13C  4C0DC0   JMP  $C00D
151  C13F  20CFFF   JSR  $FFCF
152  C142  C90D     CMP  #$0D
153  C144  D027     BNE  $C16D
154  C146  A900     LDA  #$00
155  C148  8590     STA  $90
156  C14A  2065C0   JSR  $C065
157  C14D  A908     LDA  #$08
158  C14F  85BA     STA  $BA
159  C151  20B4FF   JSR  $FFB4
160  C154  A96F     LDA  #$6F
161  C156  85B9     STA  $B9
162  C158  2096FF   JSR  $FF96
163  C15B  20A5FF   JSR  $FFA5
164  C15E  2490     BIT  $90
165  C160  7005     BVS  $C167
166  C162  20D2FF   JSR  $FFD2
167  C165  D0F4     BNE  $C15B
168  C167  20ABFF   JSR  $FFAB
169  C16A  4C0DC0   JMP  $C00D
170  C16D  C924     CMP  #$24
```

```
171  C16F  F01D     BEQ  $C18E
172  C171  48       PHA
173  C172  A908     LDA  #$08
174  C174  85BA     STA  $BA
175  C176  20B1FF   JSR  $FFB1
176  C179  A96F     LDA  #$6F
177  C17B  85B9     STA  $B9
178  C17D  2093FF   JSR  $FF93
179  C180  68       PLA
180  C181  20A8FF   JSR  $FFA8
181  C184  20CFFF   JSR  $FFCF
182  C187  C90D     CMP  #$0D
183  C189  D0F6     BNE  $C181
184  C18B  20AEFF   JSR  $FFAE
185  C18E  4C0DC0   JMP  $C00D
186  C191  2033C1   JSR  $C133
187  C194  20FEC0   JSR  $C0FE
188  C197  90F5     BCC  $C18E
189  C199  8D27C2   STA  $C227
190  C19C  2033C1   JSR  $C133
191  C19F  20FEC0   JSR  $C0FE
192  C1A2  90EA     BCC  $C18E
193  C1A4  8D2AC2   STA  $C22A
194  C1A7  2049C2   JSR  $C249
195  C1AA  AD0002   LDA  $0200
196  C1AD  C901     CMP  #$01
197  C1AF  F01E     BEQ  $C1CF
198  C1B1  A931     LDA  #$31
199  C1B3  20EDC1   JSR  $C1ED
200  C1B6  A20D     LDX  #$0D
201  C1B8  20C6FF   JSR  $FFC6
202  C1BB  A200     LDX  #$00
203  C1BD  20CFFF   JSR  $FFCF
204  C1C0  9DE0C2   STA  $C2E0,X
205  C1C3  E8       INX
206  C1C4  D0F7     BNE  $C1BD
207  C1C6  20CCFF   JSR  $FFCC
208  C1C9  206EC2   JSR  $C26E
209  C1CC  4C0DC0   JMP  $C00D
210  C1CF  202CC2   JSR  $C22C
211  C1D2  A20D     LDX  #$0D
212  C1D4  20C9FF   JSR  $FFC9
213  C1D7  A200     LDX  #$00
214  C1D9  BDE0C2   LDA  $C2E0,X
215  C1DC  20D2FF   JSR  $FFD2
216  C1DF  E8       INX
217  C1E0  D0F7     BNE  $C1D9
218  C1E2  20CCFF   JSR  $FFCC
219  C1E5  A932     LDA  #$32
220  C1E7  20EDC1   JSR  $C1ED
221  C1EA  4CC9C1   JMP  $C1C9
222  C1ED  8D20C2   STA  $C220
223  C1F0  A20F     LDX  #$0F
224  C1F2  AD27C2   LDA  $C227
225  C1F5  2078C2   JSR  $C278
226  C1F8  8E27C2   STX  $C227
227  C1FB  8D28C2   STA  $C228
```

```
228  C1FE  AD2AC2   LDA  $C22A
229  C201  2078C2   JSR  $C278
230  C204  8E2AC2   STX  $C22A
231  C207  8D2BC2   STA  $C22B
232  C20A  A20F     LDX  #$0F
233  C20C  20C9FF   JSR  $FFC9
234  C20F  A200     LDX  #$00
235  C211  BD1FC2   LDA  $C21F,X
236  C214  20D2FF   JSR  $FFD2
237  C217  E8       INX
238  C218  E00D     CPX  #$0D
239  C21A  D0F5     BNE  $C211
240  C21C  4CCCFF   JMP  $FFCC
241  C21F  5531     EOR  $31,X
242  C221  3A       BYT  $3A
243  C222  3133     AND  ($33),Y
244  C224  203020   JSR  $2030
245  C227  00       BRK
246  C228  00       BRK
247  C229  200000   JSR  $0000
248  C22C  A20F     LDX  #$0F
249  C22E  20C9FF   JSR  $FFC9
250  C231  A200     LDX  #$00
251  C233  BD41C2   LDA  $C241,X
252  C236  20D2FF   JSR  $FFD2
253  C239  E8       INX
254  C23A  E008     CPX  #$08
255  C23C  D0F5     BNE  $C233
256  C23E  4CCCFF   JMP  $FFCC
257  C241  42       BYT  $42
258  C242  2D5020   AND  $2050
259  C245  3133     AND  ($33),Y
260  C247  2030A9   JSR  $A930
261  C24A  0F       BYT  $0F
262  C24B  A8       TAY
263  C24C  A208     LDX  #$08
264  C24E  20BAFF   JSR  $FFBA
265  C251  A900     LDA  #$00
266  C253  20BDFF   JSR  $FFBD
267  C256  20C0FF   JSR  $FFC0
268  C259  A90D     LDA  #$0D
269  C25B  A8       TAY
270  C25C  A208     LDX  #$08
271  C25E  20BAFF   JSR  $FFBA
272  C261  A901     LDA  #$01
273  C263  A26D     LDX  #$6D
274  C265  A0C2     LDY  #$C2
275  C267  20BDFF   JSR  $FFBD
276  C26A  4CC0FF   JMP  $FFC0
277  C26D  23       BYT  $23
278  C26E  A90D     LDA  #$0D
279  C270  20C3FF   JSR  $FFC3
280  C273  A90F     LDA  #$0F
281  C275  4CC3FF   JMP  $FFC3
282  C278  A230     LDX  #$30
283  C27A  38       SEC
284  C27B  E90A     SBC  #$0A
```

```
285  C27D  9003     BCC  $C282
286  C27F  E8       INX
287  C280  B0F9     BCS  $C27B
288  C282  693A     ADC  #$3A
289  C284  60       RTS
290  C285  0D4449   ORA  $4944
291  C288  53       BYT  $53
292  C289  4B       BYT  $4B
293  C28A  2D4D4F   AND  $4F4D
294  C28D  4E4954   LSR  $5449
295  C290  4F       BYT  $4F
296  C291  52       BYT  $52
297  C292  205631   JSR  $3156
298  C295  2E3098   ROL  $9830
299  C298  38       SEC
300  C299  E908     SBC  #$08
301  C29B  A8       TAY
302  C29C  2062C0   JSR  $C062
303  C29F  A912     LDA  #$12
304  C2A1  20D2FF   JSR  $FFD2
305  C2A4  A208     LDX  #$08
306  C2A6  B9E0C2   LDA  $C2E0,Y
307  C2A9  297F     AND  #$7F
308  C2AB  C920     CMP  #$20
309  C2AD  B004     BCS  $C2B3
310  C2AF  A92E     LDA  #$2E
311  C2B1  D003     BNE  $C2B6
312  C2B3  B9E0C2   LDA  $C2E0,Y
313  C2B6  20D2FF   JSR  $FFD2
314  C2B9  A900     LDA  #$00
315  C2BB  85D4     STA  $D4
316  C2BD  C8       INY
317  C2BE  CA       DEX
318  C2BF  D0E5     BNE  $C2A6
319  C2C1  A992     LDA  #$92
320  C2C3  4CD2FF   JMP  $FFD2
321  C2C6  AD0102   LDA  $0201
322  C2C9  D006     BNE  $C2D1
323  C2CB  CC0402   CPY  $0204
324  C2CE  B001     BCS  $C2D1
325  C2D0  60       RTS
326  C2D1  68       PLA
327  C2D2  68       PLA
328  C2D3  4C0DC0   JMP  $C00D
329  C2D6  2065C0   JSR  $C065
330  C2D9  A93A     LDA  #$3A
331  C2DB  A23E     LDX  #$3E
332  C2DD  4CEBC0   JMP  $C0EB
```

I MESSAGGI DI ERRORE DISCO

Riportiamo qui la serie di messaggi di errore disco, le possibili cause e gli eventuali rimedi.
Per un' accurato esame degli errori e' necessario consultare le tecniche di registrazione e rilettura dati esposte nel citato volume :LE PERIFERICHE COMMODORE.
Ricordiamo che e' bene cercare di gestire SEMPRE anche da programma il controllo degli errori.

SOMMARIO DEI MESSAGGI DI ERRORE DEL DOS

0 OK, nessun errore
1 Risposta per il file cancellato.Non e' un errore
2-19Non usati
20 Intestazione non registrata
21 Carattere di sincronizzazione non registrato.
22 Blocco di dati assente
23 Errore di checksum in un blocco di dati
24 Errore di decodifica di byte
25 Errore di verifica in scrittura
26 Si e' cercato di scrivere su un disco protetto.
27 Errore di checksum nella testata
28 Blocco dati troppo lungo
29 Identificatore del disco errato
30 Errore di sintassi generico
31 Comando errato
32 Linea lunga
33 Nome del file illegale
34 File non assegnato
39 File di comando non trovato
50 Record inesistente

51 Overflow sul record
52 File troppo grande
60 File aperto per scrittura
61 File non aperto
62 File inesistente
63 File esistente
64 File inesatto
65 Blocco non disponibile
66 Traccia o settore illegali
67 Come sopra ma di sistema
70 Nessun canale disponibile
71 Errore nella Directory
72 Disco o directory piene
73 Messagio relativo all' alimentazione
74 Unita' non pronta

DESCRIZIONE DEI MESSAGGI DI ERRORE DOS

I messaggi di errore con numero inferiore a 20 dovrebbero essere ignorati con l' eccezione del messaggio 01 che fornisce informazioni sul numero di files cancellati con il comando SCRATCH.

20 READ ERROR

Non e' stata trovata l' intestazione del blocco.Il controller del disco non e' in grado di individuare l' intestazione del richiesto blocco di dati.
Questo errore puo' essere causato da un numero di settore illegale oppure l' intestazione e' andata distrutta.

21 READ ERROR

Mancanza di carattere di sincronismo.
Il controller del disco non riesce ad individuare il carattere di sincronismo nella traccia desiderata.
Puo' essere dovuto ad un disallineamento della testina, oppure non e' stato inserito il dischetto o non e' stato formattato o e' comunque sciupato.
Spesso l' errore indica anche un guasto Hardware dell' unita' a disco.

22 READ ERROR

Blocco di dati assente
E' stato richiesto al controller di leggere un blocco di dati che non e' stato in precedenza scritto in modo appropriato.
Questo errore avviene in corrispondenza di comandi BLOCK ed indica una richiesta illegale di traccia o settore.

23 READ ERROR

Errore di CHECKSUM in un blocco di dati. Indica la presenza di un errore in uno o piu' Bytes del blocco di dati.
Il blocco e' stato letto nella memoria del Sistema Operativo del disco (DOS), ma al controllo la somma di prova e' risultata errata.
Questo errore puo' esere dovuto anche ad una messa a terra errata dell' unita'.

24 READ ERROR

Errore di decodifica di byte.
I dati o l' intestazione sono stati letti nella memoria DOS, ma e' stato trovato un errore HARDWARE a causa di un errata configurazione di bit nel byte di dati.
Anche questo errore puo' derivare, come il precedente da un errata messa a terra.

25 WRITE ERROR

Errore di verifica di scrittura.
Questo messaggio viene generato se il controller individua una mancata corrispondenza fra quanto scritto e quanto contenuto nella memoria del DOS.

26 WRITE PROTECT ON

Scrittura su disco protetto.
Si ottiene questo messaggio tentando di scrivere su un dischetto con la protezione di scrittura attivata.
Cioe' quando la finestrella sulla destra del dischetto e' coperta.

27 READ ERROR

Errore di controllo somma (CHEKSUM) nella testata.
Il controller ha trovato un errore nella testata del blocco di dati richiesto.
Il blocco non e' stato correttamente letto nella memoria DOS.
Anche questo tipo di errore puo' indicare problemi HARDWARE derivanti da una errata messa a terra.

28 WRITE ERROR

Blocco di dati troppo lungo.
Il controller, dopo aver scritto un blocco di dati, cerca di individuare il carattere di sincronismo della intestazione del blocco successivo.
Se tale carattere non viene individuato entro un certo periodo di tempo e' generato il messaggio di errore.
L' errore puo' essere causato da una cattiva formattazione del disco, da un guizzo di tensione durante una fase di registrazione o da altro guasto HARDWARE.
I dati si estendono al blocco successivo che pertanto e' privo del carattere di sincronismo.

29 DISK IS MISMATCH

Errore frequente quando si tenti di usare un dischetto che non e' stato inizializzato.
Questo messaggio appare anche quando il dischetto ha una testata errata.

30 SYNTAX ERROR

Errore di tipo generale.
Il DOS non riesce ad interpretare l' ordine inviato sul canale di comando.
Generalmente questo errore e' provocato da un numero illegale di file,da nomi o configurazioni non ammessi.
Un esempio puo' essere nell' usare due nomi di files contemporaneamente con il comando COPY.

31 SYNTAX ERROR

Comando errato
Il DOS non riconosce il comando.
Si ricordi che il comando deve iniziare nella prima posizione della linea.

32 SYNTAX ERROR

Linea troppo lunga.
Il comando inviato supera la lunghezza massima di 58 caratteri.

33 SYNTAX ERROR

Nome di file non legale.
In un comando di LOAD, SAVE o VERIFY sono stati usati dei parametri illegali.

34 SYNTAX ERROR

File non assegnato.
E' stato omesso il nome del file oppure il DOS non lo riconosce.
Spesso nel comando mancano gli apici o i due punti.

39 SYNTAX ERROR

Comando non valido.
Questo errore puo' verificarsi se l' ordine inviato al canale di comando (indirizzo secondario 15) non e' interpretabile dal DOS.

50 RECORD NOT PRESENT

Record inesistente.
Questo messaggio viene generato quando si tenti una lettura al di la' dell' ultimo record tramite una operazione di INPUT# o di GET#.
Questo messaggio puo' anche aver luogo dopo il posizionamento ad un record al di la' della fine di un file quando si usino i files relatives.
Se l' intenzione era quella di espandere il file aggiungendo un nuovo record (con un comando PRINT#), il messaggio di errore puo' essere ignorato.

Le operazioni di INPUT o di GET non devono essere eseguite dopo che questo errore e' stato scoperto, se prima non si sia provveduto ad un riposizionamento.

51 OVERFLOW IN RECORD

Se l' istruzione PRINT# va oltre la dimensione del record l' informazione viene troncata.
Poiche' il ritorno carrello che serve come carattere terminatore del record deve essere contabilizzato nella dimensione del record stesso, questo messaggio di errore puo' aver luogo se il numero totale di caratteri nel record (includendo quindi anche il ritorno carrello finale) eccede la lunghezza precedentemente assegnata.

52 FILE TOO LARGE

File troppo grande.
Quando si presenta questo messaggio nel posizionamento su un record di un file relative cio' significa che si e' avuto un OVERFLOW su disco.

60 WRITE FILE OPEN

Si ha questo messaggio quando un file che era stato aperto per scriverci non e' stato richiuso e si tenta di leggerlo.

61 FILE NOT OPEN

Si ha questo messaggio quando si tenti di accedere ad un file in lettura o in scrittura e lo stesso file per il DOS non risulti aperto.
Non sempre avremo pero' questa segnalazione di errore. In qualche caso il messaggio non viene generato ed il comando di accesso e' semplicemente ignorato, e cio' e bene ricordarlo in fase di scrittura delle eventuali routines di controllo dei programmi di accesso ai dischi.

62 FILE NOT FOUND

Il file non e' stato trovato.
Il file richiesto non e' stato trovato nel drive indicato o perche' non esiste o perche' e' distrutto.

63 FILE EXISTS

Avverte che sul dischetto esiste gia' un file con lo stesso nome.

64 FILE TYPE MISMATCH

Il tipo di file specificato nella richiesta non coincide

con quello presente nella Directory.

65 NO BLOCK

Tale messaggio si puo' presentare nell' uso di un comando di BLOCK-ALLOCATE (abbreviato B-A).
Sta ad indicare che il blocco da allocare e' gia' stato allocato in precedenza.
I parametri stanno ad indicare traccia e settore del blocco disponibili piu' in alto, cioe' con un numero maggiore.
Se i parametri riportati sono 0 cio' significa che tutti i blocchi di numero piu' alto sono stati utilizzati.

66 ILLEGAL TRACK AND SECTOR

Sta ad indicare che il Sistema Operativo del Disco ha tentato di accedere ad una traccia o ad un settore che non esistono nel formato utilizzato.
Puo' anche rivelare dei problemi di lettura del puntatore al blocco successivo.

67 ILLEGAL SYSTEM T(track) OR S(sector)

Questo speciale messaggio di errore indica una traccia o un settore del sistema non legali.

70 NO CHANNEL(available)

Nessun canale richiesto e' disponibile.
Quindi o il canale richiesto non e' disponibile o tutti i canali sono gia' impegnati.

Come abbiamo visto il DOS consente di aprire solo un numero di canali limitato anche dal tipo di accesso.
Ricordiamo che possono essere aperti contemporaneamente al massimo 5 files sequenziali o 6 files ad accesso diretto.

71 DIR ERROR

Errore nella Directory.
La BAM (BLOCK AVAILABILITY MAP) non puo' essere accoppiata con il contatore interno.
L' errore puo' essere generato per problemi di allocazione della BAM stessa o quando si sia sovrascritto sulla BAM nella memoria del DOS.
Per correggere questo problema e' necessario reinizializzare il dischetto per riposizionare giustamente la BAM nella memoria DOS.
Alcuni files aperti devono essere chiusi con la fase di correzione.

72 DISK FULL

Il dischetto e' stato completamente riempito oppure , anche con spazio ancora su disco, si e' pero' utilizzta l' intera capacita' del Directory di memorizzare i nomi di 144 files.

73 DOS MISMATCH

La versione del DOS dell' unita' a dischi 1540 e 1541 e' la CBM DOS 2.6.
Le versioni DOS 2.6 e DOS 1.0 (cioe' quelle delle unita' CBM 2040 e 3040) sono compatibili in lettura ma

non in scrittura.
I dischi cioe' possono essere letti con intercambiabilita' con i due DOS, ma un disco formattato con una delle due versioni non puo' essere scritto con l' altra perche' la disposizione varia.
Questo errore viene segnalato quando si tenti di scrivere su un disco che e' stato formattato in un' altra versione.
Purtroppo pero' i DOS 2.6 e 2.5 (cioe' quelli presenti sulle unita' Commodore 8050 e 8250) non sono compatibili neppure in lettura.
Ricordiamo che questo messaggio puo' apparire anche dopo l' accensione.

74 DRIVE NOT READY

Unita' non pronta nel senso che all' interno manca il dischetto.

75 FORMAT SPEED ERROR

Questo messaggio d' errore si ha solo con l' unita' 8250. Indica uno scostamento di velocita' dalla norma durante la fase di formattazione.

INDICE

COGNOME........................
NOME ....
INDIRIZZO......................
CAP/CITTA'.....................

DESIDERO RICEVERE:

☐ GRATUITAMENTE IL CATALOGO EVM ED ESSERE INSERITO NELLA MAILING LIST CON TUTTI GLI AGGIORNAMENTI

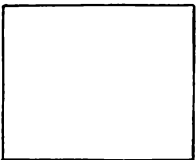
☐ IL DISCO CONTENENTE I PRÒGRAMMI INPUT*, SPOOLING, DISK MONITOR AL PREZZO DI LIRE 20.000 (IVA COMPRESA)

☐ IL LIBRO LE PERIFERICHE COMMODORE AL PREZZO SPECIALE DI L. 20.000 (SC 20%)

☐ IL LIBRO IL SISTEMA OPERATIVO DEL CBM64 CON NASTRO CON I PROGRAMMI :MONITOR, ASSEMBLER, DISASSEMBLER AL PREZZO SPECIALE DI 30.000 (SC 20 %)

☐ IL CORSO COMPLETO DI ASSEMBLER PER CBM64 CON REELATIVO PROGRAMMA AL PREZZO SPECIALE DI L. 34.000 (SC. 10%)

☐EFFETTTUATE LA SPEDIZIONE IN CONTRASSEGNO. PAGHERO' AL POSTINO PIU' UN CONTRIBBUTO DI LIRE 4.000 PER SPESE POSTALI

☐ALLEGO L' IMPORTO PIU' LIRE 1800 PER CONTRIBUTO SPESE POSTALE

E.V.M. Computers
Via Marconi 9/A - Loc.Muraccio -
MONTEVARCHI (AR) 52025